System z

Hardware Management Console Web Services API Version 2.12.0

SC27-2617-01

System z

Hardware Management Console Web Services API Version 2.12.0

SC27-2617-01

Note:
Before using this information and the product it supports, read the information in "Safety" on page xv, Appendix B, "Notices," on page 685, and *IBM Systems Environmental Notices and User Guide*, Z125–5823.

This edition, SC27-2617-01, applies to the IBM zEnterprise System servers and all follow-on IBM System z servers. This edition replaces SC27-2617-00.

There might be a newer version of this document in a **PDF** file available on **Resource Link**. Go to http://www.ibm.com/servers/resourcelink and click **Library** on the navigation bar. A newer version is indicated by a lowercase, alphabetic letter following the form number suffix (for example: 00a, 00b, 01a, 01b).

Contents

Figures

Tables

Safety

Safety notices

Safety notices may be printed throughout this guide. **DANGER** notices warn you of conditions or procedures that can result in death or severe personal injury. **CAUTION** notices warn you of conditions or procedures that can cause personal injury that is neither lethal nor extremely hazardous. **Attention** notices warn you of conditions or procedures that can cause damage to machines, equipment, or programs.

World trade safety information

Several countries require the safety information contained in product publications to be presented in their translation. If this requirement applies to your country, a safety information booklet is included in the publications package shipped with the product. The booklet contains the translated safety information with references to the US English source. Before using a US English publication to install, operate, or service this IBM® product, you must first become familiar with the related safety information in the *Systems Safety Notices*, G229-9054. You should also refer to the booklet any time you do not clearly understand any safety information in the US English publications.

Laser safety information

All System z® models can use I/O cards such as PCI adapters, FICON®, Open Systems Adapter (OSA), InterSystem Coupling-3 (ISC-3), or other I/O features which are fiber optic based and utilize lasers or LEDs.

Laser compliance

All lasers are certified in the US to conform to the requirements of DHHS 21 CFR Subchapter J for Class 1 or Class 1M laser products. Outside the US, they are certified to be in compliance with IEC 60825 as a Class 1 or Class 1M laser product. Consult the label on each part for laser certification numbers and approval information.

CAUTION: Data processing environments can contain equipment transmitting on system links with laser modules that operate at greater than Class 1 power levels. For this reason, never look into the end of an optical fiber cable or open receptacle. (C027)

CAUTION: This product contains a Class 1M laser. Do not view directly with optical instruments. (C028)

About this publication

This publication defines, for reference purposes, the external interface of the Hardware Management Console (HMC) Web Services Application Programming Interface (Web Services API) for IBM zEnterprise®, Version 2.12.0. This document specifies the capabilities, input and output formats, and behaviors of the Web Services API as viewed by an application external to the HMC that is leveraging that interface.

Related publications

The following publications provide information which supplements the information found within this document:

- *System z Hardware Management Console Operations Guide*, SC28-6919
- *zEnterprise System Capacity On Demand User's Guide*, SC28-2605
- *zEnterprise System Ensemble Performance Management Guide*, GC27-2607
- *zEnterprise System Ensemble Planning and Configuring Guide*, GC27-2608
- *zEnterprise System Processor Resource/Systems Manager Planning Guide*, SB10-7156
- *zEnterprise System Support Element Operations Guide*, SC28-6920
- *z/VM CP Planning and Administration Guide*, SC24-6178
- *z/VM CP Commands and Utility Reference*, SC24-6175
- *zEC12 Installation manual for Physical Planning*, GC28-6914-00
- *zBX Model 003 Installation Manual for Physical Planning*, GC27-2619-00
- *zBX Model 002 Installation Manual for Physical Planning*, GC27-2611-03

Revision bars

A technical change to the text is indicated by a vertical line to the left of the change.

For more information about what has changed since the last publication, see "Summary of API version updates" on page 5.

Accessibility

This publication is in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties using this PDF file you can request a web-based format of this publication. Go to Resource Link® at http://www.ibm.com/servers/resourcelink and click **Feedback** from the navigation bar on the left. In the **Comments** input area, state your request, the publication title and number, choose **General comment** as the category and click **Submit**. You can also send an email to reslink@us.ibm.com providing the same information.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. Send your comments by using Resource Link at http://www.ibm.com/servers/resourcelink. Click **Feedback** on the navigation bar on the left. You can also send an email to reslink@us.ibm.com. Be sure to include the name of the book, the form number of the book, the version of the book, if applicable, and the specific location of the text you are commenting on (for example, a page number, table number, or a heading).

Chapter 1. Introduction

This chapter provides an overview of IBM zEnterprise Unified Resource Manager (zManager) APIs, how to enable and access them, and considerations for compatibility.

Overview

The Unified Resource Manager (zManager) is a collection of advanced hardware and virtualization management functions delivered as System z firmware. The functions of zManager are implemented as a cooperating set of components hosted on the Hardware Management Console (HMC), the Support Element (SE), the blades of a zEnterprise BladeCenter® Extension (zBX) , and as extensions to z/VM®. It provides a uniform, integrated and workload-oriented administrative model for the heterogeneous computing configuration provided by a zEnterprise system. The functions provided by zManager include:

- Hardware inventory, initialization, configuration, monitoring and problem analysis for the components of a System z CPC, including both the traditional System z computing resources as well as the zBX.
- Firmware installation and update for the HMC, SE, traditional CPC components, zBX infrastructure, Intra-ensemble data network (IEDN) elements, and zBX blades.
- Operational control and energy management for these hardware elements.
- Configuration of function-specialized workload accelerators available as blade optimizers in the zBX.
- Provisioning, configuration, control and monitoring of virtualized computing systems (virtual servers) on the firmware-managed IBM blade and z/VM environments.
- Secure management of the IEDN through the provisioning and control of IEDN virtual networks.
- Automatic and workload-oriented performance optimization of the heterogeneous, virtualized environment.

The HMC serves as the administrative access point for zManager. In that capacity, the HMC provides a web-based, remote-able graphical user interface (UI) to make the zManager functions available to users. In addition, it hosts the implementation of zManager Web Service API (Web Services API) that is described in this document.

The Web Services API is a web-oriented programming interface that makes the underlying zManager capabilities available for use by higher level management applications, system automation functions, or custom scripting. The functions that are exposed through the API support several important usage scenarios in virtualization management, including resource inventory, provisioning, monitoring, automation and workload-based optimization among others.

Components of the API

The Web Services API consists of two major components. Both components are accessed by client applications by establishing TCP/IP network connections with the HMC.

Web services interface

The web services interface is a request-and-response oriented programming interface by which client applications obtain information about the system resources managed by zManager, and by which those applications can perform provisioning, configuration or control actions on those resources.

As is the case for any web-oriented interface, client applications interact with this interface by means of the Hypertext Transfer Protocol (HTTP), an application protocol that flows over TCP/IP socket connections. Client applications request operations by forming and sending text-oriented request messages as defined by HTTP, and the Web Services API responds with text-oriented HTTP response messages. The use of HTTP makes the API client-programming-language neutral, and thus accessible to a

wide variety of client applications. Client applications can be developed in programming languages such as Java™, or in scripting languages such as Perl or Python that include extensive support for performing HTTP operations.

The design of the API's mapping to HTTP has been influenced by the Representational State Transfer (REST) style of interface design. The manageable resources of the system are associated with and identified by durable URIs, and the basic get, update, create and delete operations on those manageable resources are mapped directly to the HTTP GET, PUT, POST and DELETE methods. Request and response data is provided using JavaScript Object Notation (JSON), a simple, open and portable transfer representation. Mapping the functions of the API to HTTP in this way simplifies client application development and allows access to the API without the need for extensive client side tooling or libraries as is often the case in other approaches to web services interface design.

Broadly speaking, the web service interface provides two categories of operations:

- Resource (or object) oriented operations, in which a particular request is targeted at a single manageable resource instance and typically affects just that single resource instance. The majority of the API has this orientation, for example providing functions for interacting with the virtual servers, virtualization hosts, virtual networks and workloads of the system.
- Service oriented operations, in which a particular request operates across many or all manageable resources of the system. The service-oriented operations are provided to support usage scenarios that cannot be accomplished efficiently using an object-by-object sequence of individual requests. The operations provided by the Metrics and Inventory services of the API are examples of service-oriented operations.

Asynchronous notification facility

The web services interface described above is useful to satisfy many usage scenarios, particularly those in which the client application's interest in and interaction with zManager is focused on performing a short-term task. In these kinds of applications (typical of automation or simple provisioning), the client application forms a request, gets a response, processes the response and then "forgets" about the zManager resource it interacted with. That is, the application does not attempt to retain (or cache) information about zManager resources long term and then keep that cache up to date.

However, more sophisticated management applications, including those for discovery, monitoring and advanced provisioning, are not single-request-and-forget with respect to their interest in zManager. Rather, such applications have a need to obtain and retain (i.e., cache) information about the inventory, configuration and status of many zManager resources, and to keep that cached information up to date.

In order to support these more sophisticated applications, the Web Services API provides an asynchronous notification facility by which zManager can inform interested client applications about changes to the resources managed by zManager.

The API's asynchronous notification facility is designed around the Java Message Service (JMS), an open, standard framework and API for sending messages between two or more applications.

Enabling and accessing the API

The Web Services API is provided on an HMC that is running with firmware version 2.11.1 or later. The API can be used to query, configure and control Central Processing Complexes (CPCs) containing a Support Element (SE) that is running with firmware version 2.11.1 or higher.

By default, the Web Services API is disabled on the HMC. When disabled, the HMC internal firewall is configured to prohibit connections to any of the TCP/IP ports used by the API. When in this state, requests to connect to the API network ports are completely ignored by the HMC without a connection-refused response.

The Web Services API can be enabled and the scope of access to it configured using the **Customize API Settings** task in the HMC UI. This task, which previously provided configuration settings for the SNMP APIs, has been extended with new controls for the Web Services API as well.

The **Customize API Settings** task allows an installation to enable the API via an overall enabled/disabled setting. When enabled, the HMC internal firewall is reconfigured to allow access to the relevant network ports. When the API is enabled with default settings, the HMC allows connections to the API functions from client applications accessing the HMC from any TCP/IP address. For additional security, an installation can configure the HMC to permit connections to the API ports only from selected network addresses or subnets. These addresses or subnets are specified by the **Customize API Settings** task as well. If specified, these connection restrictions are enforced by the HMC internal firewall.

In addition to the overall enablement on/off control and the optional client network address filtering, access to the API is further secured by the requirement for per-user authorization.

The HMC **User Profiles** and **Manage Users Wizard** tasks define access and other characteristics of an HMC user. These tasks have been extended to provide a new property (**Allow access to management interfaces**) to indicate whether a particular HMC user is to be permitted to use the API or not. By default, this setting is disabled for an HMC user profile and thus attempts to establish an API session by that user are rejected. The installation can use the **Customize API Settings**, **User Profiles**, or **Manage Users Wizard** tasks of the HMC to set this property for one or more HMC users and thus allow those users to access the API.

Once a user is permitted to establish API sessions, its actions within those sessions are subject to the HMC's access control model, as is described in the section that follows.

Authentication and access control

The HMC provides a built-in access control model in which an HMC user authenticates itself to the HMC to establish its identity, and then based on that identity is permitted or denied the ability to perform certain operations as specified by the access control configuration. These operations, and the objects on which they are permitted, are managed with object and task/action permissions that are grouped into roles that are assigned to HMC users. Roles are managed with the **Customize User Controls** task on the HMC. Roles are assigned to users with the **User Profiles** or **Manage Users Wizard** tasks.

Use of the Web Services API is subject to the same access control policy as is used for UI operations.

Establishing an API session with the HMC requires the initiating application to provide a valid HMC logon ID and corresponding password in order to authenticate and establish the identity under which its requests will be performed. (See “Logon” on page 39 for more information.) The API requires the use of SSL connections so that these login credentials can be flowed securely. The user credentials are validated by the HMC in the same way they are validated for a logon to the UI, either via the HMC's built-in user registry or by use of an LDAP directory server.

Once a client application has established an API session, its ability to access various managed object instances and the operations that can be performed on those instances is regulated based on the identity associated with the API session and the access control policy configured in the HMC for those managed object instances. Access control requirements vary based on the class of managed object and the operation for the managed object. These access control requirements for API actions mirror the requirements for corresponding tasks in the HMC user interface. Details on the authorization requirements for an operation are specified in the description of that operation.

Alternate HMC considerations

A zEnterprise ensemble is managed by a pair of HMCs that operate in a primary/alternate configuration rather than by a single HMC. At any point in time, one HMC of the pair is designated as the primary HMC and has active management responsibility for the resources of the ensemble. The other HMC is

designated as the alternate HMC, and when in this role acts only as a standby for the primary HMC and mirrors configuration updates from the primary in case the primary fails, but does not otherwise perform active management.

The Web Services API can and should be enabled on both the primary and alternate HMCs of the pair.

However, client applications should generally connect only with the primary HMC for API purposes. Because the alternate HMC is not performing active management, it is unable to perform the management actions implied by API requests, and thus most API operations are designed to be rejected if directed to the alternate HMC (with HTTP status code 404, reason code 3). The API operations supported on the alternate HMC are limited to those that control the alternate HMC function itself. The description of an operation specifically indicates that it is supported on the alternate HMC if this is the case.

Compatibility

The capabilities of the Web Services API will evolve as additional management functionality is added to zManager. Over time, this evolution could result in a mixture of HMC and client application versions coexisting in a customer environment. The principles and guidelines outlined in this section are intended to maximize the compatibility and interoperability among HMC and client applications in such a mixed environment.

API versioning

Since the functionality of the Web Services API may evolve over time, each functional level of the API is identified by a version number. This version number is represented in major.minor form, with the initial version of the API designated as version 1.1.

The API version offered by an HMC can be determined before API logon by using the Query API Version operation (GET /api/version). The version number of the API is also provided in the response from the Logon operation.

Enhancements to the API specification that maintain compatibility with previous versions (see principles below) are indicated by incrementing the minor portion of the version number. So, for example, the first set of compatible changes to the API would be designated as version 1.2, following the initial 1.1 version.

Because the minor versions within a major version stream (e.g. the 1.x versions) are considered compatible, the HMC always offers and behaves according to the latest minor version of the API specification it supports. That means, for example, the API does not offer any facility by which a client can request version 1.1 behaviors on an HMC that offers version 1.2 level of functionality.

While reasonable effort will be made to preserve compatibility, it may become necessary to make changes to zManager (and thus the API) that do not maintain compatibility with the previous version. If this occurs, the introduction of this new (incompatible) behavior is indicated by incrementing the major part of the version number, and starting the minor part of the version number again at 1. The first such version would thus be identified as version 2.1.

Allowable changes within a major version

The following kinds of changes to the API specification are allowable within a major version, and thus result in changes to the minor but not major parts of the API version number.

- Adding new object classes or new operations on existing object classes.
- Adding new properties to the data model of an existing object class.
- Changing existing properties of an existing object class from read-only or mutable to writable using the API.
- Adding new URIs and operations related to those URIs.

- Adding new optional query parameters to existing URIs where the behavior in the absence of this query parameter is unchanged.
- Adding new optional fields into input bodies where the behavior in the absence of these new fields is unchanged.
- Adding new fields to the response bodies of existing operations.
- Adding additional header or body fields to existing notification messages.
- Adding data for new classes of objects to the results provided by the Inventory service.
- Generating new types of notification messages.
- Generating property change notifications for new properties, or for existing properties that did not provide those notifications previously.
- Adding new enumeration values to enumeration-type fields returned in response bodies without removing or changing the meaning of any existing enumeration values.
- Adding new error status and reason codes.
- Adding new metric groups.
- Adding new metric fields to the end of existing metric groups.

Requirements on client applications

In order for a client application to correctly interoperate with an HMC that may be offering a higher minor version of the API, client applications must be designed and developed following the simple principle of "ignore what you don't understand" when interpreting responses or messages received from the HMC. This is necessary because the principles of allowable changes specified in the preceding section allow new fields to be added to preexisting responses or messages.

More specifically, a client application must:

- Ignore, without error, any field in a response body that is not recognized by the application.
- Ignore, without error, any header or body field in a notification message that is not recognized by the application.
- Ignore, without error, any notification message of an unrecognized type that may be received by the application.
- Ignore, without error, any object appearing in the response to an Inventory request that are of a class not recognized by the application.
- Tolerate receiving a value in a field with an enumeration data type that is an unexpected value. If the application is attempting to display this field, it might consider mapping the unrecognized enumeration value to some value indicating "other" or "unknown".
- Ignore, without error, extra values provided in a row of metric group data that reside beyond the last field currently expected by the application.

These conditions can arise as a result of API extensions that are considered allowable within a given major version. Following the "ignore what you don't understand" principle prepares a client application to tolerate these API additions should they occur.

Summary of API version updates

The following functions were introduced in the respective API version:

Table 1. Summary of updates by API version number

API Ver	Description	HMC MCL	SE MCL
	HMC/SE Version 2.11.1		
1.1	Added reason codes 0, 105 and 108 as possible HTTP status code 400 (Conflict) error conditions reported by the **Migrate Virtual Server** operation.	N48180.278	N48168.275

Table 1. Summary of updates by API version number (continued)

API Ver	Description	HMC MCL	SE MCL
1.1	Changed the **backing-virtualization-host-storage-resource** property of the Virtual Server data model (in the fullpack-virtual-disk nested object) from a read-only property to a writeable property.	N48180.287	None
1.1	Increased the maximum request body size for the **Import Storage Access List** operation to 1 MB, and increased the maximum request body size for most other operations to 64KB.	N48180.288	N48168.294
1.1	Added the **cores-per-processor** property to the Blade object data model (read-only).	N48180.296	None
1.1	Added inventory and property-change notification support for the Virtualization Host Storage Resource object.	N48180.308	N48168.315
1.1	Added the **inventory-error-details** field and related inventory-error-info nested object to the inventory-error document returned by the **Get Inventory** operation when error condition 5 is encountered.	N48180.314	N48168.321
1.1	• Added the **properties=common** query parameter to the **Get Virtual Server Properties** operation. • Added the virtual-server-common category and power-vm-virtual-server-common, prsm-virtual-server-common, x-hyp-virtual-server-common and zvm-virtual-server-common classes to the **Get Inventory** operation.	N48180.319	N48168.325
1.1	Changed the **resources** field in the request body for the **Get Inventory** operation from required to optional.	N48180.340	None
1.1	Corrected the range checking for the **load-address** field of the **Load Logical Partition** operation so that the operation correctly supports loading from an alternate subchannel.	N48180.360	None
1.2	• Added the **Mount Virtual Media Image** operation. • Increased API version number from 1.1 to 1.2.	N48180.361	None
1.2	Corrected the format of the URI returned by the **List Members of Virtual Network** operation for a zBX TOR port to reflect the correct canonical URI format for zBX TOR port elements.	N48180.363	None
1.2	Added the **power-vm-partition-id** property to the Virtual Server object data model for PowerVM virtual servers (read-only)	N48180.363	N48168.378
1.2	Added HTTP status code 409 (Conflict) as a possible error condition for the List Virtualization Host Storage Resources and Get Virtualization Host Storage Resource Properties operations.	N48180.376	None
1.2	Added the **feature-list** property to the Virtualization Host object data model. This property is provided for virtualization hosts on all CPCs supported by the Web Services API, but the particular features provided by a given virtualization host will differ based on the release and MCL level of the CPC.	N48180.380	N48168.402
HMC/SE Version 2.12.0			

Table 1. Summary of updates by API version number (continued)

API Ver	Description	HMC MCL	SE MCL
1.3	The following extensions are provided by the HMC Web Services API for HMCs at version 2.12.0, and apply to all CPCs supported by the Web Services API: • Increased API version number from 1.2 to 1.3. • Added the **power-saving-state** property to the for BladeCenter and Blade objects data models, and added the **cpc-power-saving-state** and **zcpc-power-saving-state** properties to the CPC object data model. • Added **"not-supported"** as a possible enumeration value for the power-save-allowed and power-cap-allowed properties of BladeCenter and Blade objects, and added **"not-supported"** as a possible enumeration value of the **cpc-power-save-allowed**, **cpc-power-cap-allowed**, **zcpc-power-save-allowed** and **zcpc-power-cap-allowed** properties of the CPC object. • Added the **status** (read-only), **acceptable-status** (writeable), **perf-status** (read-only) and **compliant-perf-status** (writable) properties to the Workload Resource Group object data model. • Added **most-severe-perf-status** and **perf-status-data-points** fields, and related **perf-status-data-point** nested object to the response from the **Generate Workload Resource Groups Report** operation. • Added the **perf-policies** property to the Virtual Server object data model, and also added related virtual server performance policy nested object. • Added **"data-retrieval-error"** as a possible enumeration value for the status-detailed field in the response for the **Generate Load Balancing Report** operation. • Added an optional request body containing an optional **force** input field to the **Unmount Virtual Media** operation. • Changed the **Create Virtual Server** operation for a zVM virtual server to require the **password** field on input rather than allowing it to be optional. This change has been made to improve security. • Added HTTP status code 409 (Conflict) as a possible error response reported by the following Storage Management operations: – **Import Storage Access List** – **Create Virtualization Host Storage Resource** – **Delete Virtualization Host Storage Resource** – **Add Virtualization Host Storage Resource Paths** – **Remove Virtualization Host Storage Resource Paths** – **Discover Virtualization Host Storage Resources.**	H09182.023	H09173.028

Table 1. Summary of updates by API version number (continued)

API Ver	Description	HMC MCL	SE MCL
1.3	The following extensions are provided by the HMC Web Services API for HMCs at version 2.12.0, but apply only to CPCs with SE version 2.12.0: • Added **cp-cpu-consumption-percent**, **ifl-cpu-consumption-percent** and **other-cpu-consumption-percent** fields to the response from the **Generate Hypervisor Report** operation for zVM virtualization hosts. These new fields are provided for zVM virtualization hosts running at version 6.2 or greater. • Added the following in support of IBM zAware partitions. These changes apply only for partitions of the new **"zaware"** type: – Added **"zaware"** as a possible value of the **activation-mode** property of Logical Partition and Image Activation Profile objects. – Added the zaware-network, network-ip-info and ip-info nested objects as common nested object definitions used for new properties of Logical Partition objects. – Added the **zaware-host-name**, **zaware-master-userid**, **zaware-master-pw**, **zaware-network-info**, **zaware-gateway-info** and **zaware-dns-info** properties to the Logical Partition and Image Activation Profile object data models. – Added HTTP status 400 reason code 306 as a possible error response from the Load Logical Partition, PSW Restart, Start Logical Partition, Stop Logical Partition, and Update Image Activation Profile Properties operations when these operations are attempted on an IBM zAware partition. • Added the Cryptos and Flash Memory Adapters metric groups for CPC objects. Data entries are provided for a CPC in these metric groups if the CPC has one or more Cryptos or Flash Memory Adapters installed. • Added new **cp-cpu-time**, **ifl-cpu-time**, **zaap-cpu-time**, **ziip-cpu-time** and **icf-cpu-time** metrics to the Virtualization Host CPU and Memory metric group (for zVM virtualization hosts).	H09182.023	H09173.028
1.3	Added HTTP status code 409 (Conflict) as a possible error condition for the List Virtualization Host Storage Resources and Get Virtualization Host Storage Resource Properties operations.	H09182.062	None
1.3	Added property change support for the **unique-device-id** property of the Storage Resource object.	H09182.102	None
1.3	Added the **feature-list** property to the Virtualization Host object data model. This property is provided for virtualization hosts on all CPCs supported by the Web Services API, but the particular features provided by a given virtualization host will differ based on the release and MCL level of the CPC.	H09182.119	H09173.149

Chapter 2. Base definitions

This chapter provides basic definitions of data types, representation formats and other fundamental syntactic elements that apply across the Web Services API.

Data types

The following data types are used in the definition of the management data model, input and output parameters and notification message formats in the Web Services API.

Table 2. Primitive data types

Data type	Description
Boolean	A logical truth value: either the value **true** or the value **false**.
Byte	An integer value in the range -2^7 to (2^7)-1 (the range of a signed 8-bit integer)
Float	An IEEE 754 floating point number in the range +/-4.9E-324 to +/-3.4028235E+38. Note that, although IEEE 754 provides for representations of positive or negative Infinity and NaN, such values are not used within the API.
Long	An integer value in the range -2^63 to (2^63)-1 (the range of a signed 64-bit integer)
Integer	An integer value in the range -2^31 to (2^31)-1 (the range of a signed 32-bit integer)
Short	An integer value in the range -2^15 to (2^15)-1 (the range of a signed 16-bit integer)
String	A sequence of Unicode characters. When the number of characters in the string is bounded, the length or length range is provided in parenthesis, for example String (16) for a 16 character string, or String (0-256) for a string that may range in length from 0 (empty) to 256 characters.
String Enum	A String enumeration, i.e. a String for which the possible values are constrained to be one of a specified set of choices.
String/URI	A String that contains a URI path used to designate object instances or operations within the API.
String/IPV4 Address	A String that contains an Internet Protocol Version 4 address presented in dotted-decimal notation. Example: “`127.0.0.1`”
String/IPV6 Address	A string that contains an Internet Protocol Version 6 address presented in colon-separated-hexadecimal notation. Leading and consecutive groups of zeros may be omitted in the representation as is conventional for IPV6 addresses presented in this form. Example: “`2001:db8:85a3::8a2e:370:7334`”
Timestamp	A Long integer quantity where the value represents a date and time expressed as the number of milliseconds since midnight on January 1, 1970 UTC.

Table 3. Compound data types

Data type	Description
Array of <T>	A ordered sequence of zero or more elements each of data type <T>. An array may be empty, i.e. have no elements contained within it.
Object	A nested data structure providing a set of fields, each field having a name, data type and value. Object types do not formally have names. However, descriptions of these nested objects will often assign reference names to allow connections to be made in the documentation between points of use and definition for a given nested object.

Input and output representation

Except for a few special cases, the operations provided by the Web Services API expect their input and provide their output using a representation known as JavaScript Object Notation, or JSON for short. The JSON representation is also used within the bodies of notification messages emitted by the API. Unless some different representation is specifically mentioned in the description of an operation or message, all operations and messages should be understood to use JSON notation.

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format that defines a small set of formatting rules for the portable representation of structured data. JSON can represent four primitive types (strings, numbers, booleans, and the value null) and two structured types (objects and arrays) that together provide sufficient expressive power to represent the manageable resource configuration, state, inputs, and outputs that appear in this API.

A JSON string is a sequence of zero or more Unicode characters enclosed in quotes.

A JSON object is an unordered collection of zero or more name/value pairs (sometimes referred to in this document as fields or properties), where a name is a string and a `value` is a primitive type (string, number, boolean, or null), an array, or a nested object. Each name/value pair is represented in the form **"name"**: value and is separated from the next name/value pair by a comma. The collection of name/value pairs comprising the object is enclosed by left and right braces e.g. { `...` }).

An array is an ordered sequence of zero or more values separated from each other by commas and enclosed in left and right square brackets e.g. `[10,20,30]`). The values in the array can be primitive or structured types, i.e. arrays of objects or arrays of arrays are permitted.

The precise BNF syntax of JSON notation is not provided in this document, but can be found in the IETF information document RFC 4726, The *application/json Media Type for JavaScript Object Notation (JSON)*, July 2006. This RFC can be found in text format on the World Wide Web at:

http://www.ietf.org/rfc/rfc4627.txt

Representing API data types in JSON

The following tables define the mapping between the API data types and their corresponding representation in JSON notation.

Table 4. Primitive data types in JSON notation

API data type	JSON representation
Boolean	A JSON boolean with keywords **true** and **false**
Byte, Integer, Long, Short	A JSON number with a sign and integer component, but no fraction or exponent part.
Float	A JSON number, possibly including fraction or exponent parts. On output, values with a magnitude greater than or equal to 10^-3 and less than 10^7 are representation in floating-point format with a fraction part but not exponent part (e.g. 1.7, -32.467). Values with magnitudes outside that range are represented in scientific notation with both fraction and exponent parts (e.g. -4.23E127).
String, String Enum	Represented as a JSON string enclosed in quotes.
Timestamp	An unsigned JSON number with integer component, but no fraction or exponent part.

Table 5. Compound data types in JSON notation

Data type	Description
Array of <T>	A JSON square-bracket-enclosed array with elements represented according to the data type <T>.
Object	A JSON curly-brace-enclosed object, with the fields/properties of the nested object represented as name/value members of the object. The name of a property/field is used directly as the name part of the JSON object member, and the value of the field/property is provided as the value part of the member.

All strings in the JSON representation (object member names, and string values) are encoded in UTF-8.

Chapter 3. Invoking API operations

The Web Services API provides an extensive set of operations that client applications can invoke to obtain information about the manageable resources of the system, to change those resources' characteristics, and to take action on them. Because the API is designed using a web services orientation, these operations are accessed by means of Hypertext Transport Protocol (HTTP) protocol messages flowing across TCP/IP network connections.

Most aspects of HTTP protocol usage required to invoke API operations or receive responses apply universally across all of the operations of the API. Rather than repeat these details in the description of each and every operation, this common information is instead provided in this chapter. The material in this chapter should be considered to apply to each and every operation of the API unless the operation-specific description indicates otherwise. Thus, the information in his chapter should be consulted in conjunction with the operation-specific descriptions elsewhere in this document when determining how to invoke a specific API operation.

HTTP protocol standard

The Web Services API has been designed in accordance with the HTTP version 1.1 protocol, as defined in the W3C internet standards document *RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, June 1999*. This RFC can be found in HTML format on the World Wide Web at: http://www.w3.org/Protocols/rfc2616/rfc2616.html

The API requires that all clients interact using the HTTP/1.1 protocol. The API does not support clients that use HTTP/1.0.

Note: While the API does not specifically assume or exclude any particular client user agent, its use and interpretation of HTTP elements has been designed presuming that the client application interacting with the API is a programmatic web application client or HTTP-capable scripting client rather than a standard browser-based application.

Connecting to the API HTTP server

When the Web Services API is enabled, the HMC API HTTP server listens for SSL-based socket connections on TCP port 6794. The HMC is enabled for both the SSL version 3 and TLS version 1 protocols on this SSL port. It does not accept non-SSL connections.

The listening port for the API HTTP server is a fixed port number and is not subject to customer reconfiguration. Thus, client applications can treat this as a well-known port number rather than requiring customer input when configuring the networking parameters the client will use to connect to the HMC.

HTTP header field usage

HTTP request and response messages include elements known as headers fields (often referred to simply as headers for short) that provide request metadata. Certain headers are required or provided in all HTTP messages, while others are present in selected messages depending on content.

This section describes the use of header fields by the Web Services API.

Required request header fields

The following HTTP request headers are relevant to all request methods (GET, PUT, POST, DELETE) and are required on all API requests (except as indicated for the Logon and Query API Version operations).

HTTP header name	Rqd/Opt	Description
Host	Required	Specifies the Internet host and port number of the HMC to which the request is being directed, as obtained from the original URI given by the client application. The Web Services API enforces that this header is provided as required by the HTTP protocol, but does not check or use the value of the header in any way.
X-API-Session	Required[1]	An opaque string that provides a cryptographically strong identifier of the API session (known as a session id) under which this request is executed. This header is required on all requests that require authentication. The Login operation begins a new HMC session and includes credentials identifying the HMC user for the session. Upon successful authentication, the Login operation returns the value to be used in the **X-API-Session** header for all subsequent requests of the same session. Failure to include this header on a request requiring authentication results in status code 403 (Forbidden) with reason code 4. Specifying an invalid session id results in status code 403 (Forbidden) with reason code 5.
Note: 1. Not required on requests to the Query API Version and Logon operations since these operations can be performed before an API session has been established.		

For requests made using the HTTP PUT or POST methods, the following additional request headers are required if a request body is being provided. If an operation being requested via POST method does not require a request body, these headers can be omitted.

HTTP header name	Rqd/Opt	Description
Content-Length	Required if request body present	When used in a request, specifies the length of the request body. If omitted, the request is presumed to not contain a body. The API limits the size of request bodies in order to control usage of memory resources on the HMC. Unless a different limit is specified for a particular operation, in general the largest request body accepted by the API is 64KB. Requests with bodies that exceed this maximum are rejected with an HTTP status 413 (Request Entity Too Large) response.
Content-Type	Required if request body present	When used in a request, specifies the MIME media type of the request body contained in the request. This header is required if the **Content-Length** header is supplied and specifies a non-zero request body length, otherwise status code 400 (Bad Request) will result.

Optional request headers

The following HTTP request headers are relevant to all request methods (GET, PUT, POST, DELETE) and may be specified on these method requests but are not required. If present, they are interpreted by the API in the indicated way.

HTTP header name	Rqd/Opt	Description
Accept	Optional	Specifies the list of response MIME media types that the client application is prepared to accept for the response to the request. This header is provides for content negotiation between the client and the server in cases where the Web Services API supports multiple possible response media types for a given operation. In the current implementation, the Web Services API supports only a single response media type for each operation. For the majority of operations, that media type is JSON (**`application/json`**), but selected operations support a different media type (indicated in the descriptions of those special operations). If this header is omitted, the Web Services API responds using the (single) media type supported for the operation. If the header is included, it must allow for the single media type that the operation supports, otherwise the request will fail with HTTP status code 406 (Not Acceptable). If an operation is extended to support multiple media types, compatibility will be maintained for existing clients that request the operation without specifying an **Accept** header.
X-Audit-Id	Optional	A string that provides additional client identity information that is included in all audit records created for this request, in addition to the API user's HMC login identity. This header is intended to provide improved audit logging in the case of clients that make requests on behalf of multiple upstream users while logged into the API under a single HMC login identity. Such clients should provide the identity of their upstream user in this header so that the requests of different upstream users can be distinguished in the HMC audit logs. The HMC will use up to the first 64 characters of information from this header if present, and silently ignore the remainder of the header's value if it is longer than 64 characters.
X-Client-Correlator	Optional	A string that provides diagnostic information pertaining to this request that is of significance to the client, such as a client request number or the like. The HMC will record this information in selected diagnostic trace or log data it collects so as to allow better cross-correlation of this information with similar information maintained by the client. This data supplied in this header is intended to assist in product problem determination and does not otherwise affect the operation of the API. The HMC will use up to the first 64 characters of information from this header if present, and silently ignore the remainder of the header's value if it is longer than 64 characters.

Standard response headers

The following HTTP response headers are always provided in the response to all requests.

HTTP header name	Description
Date	The date and time, from the perspective of the HMC's clock, at which the response message was generated. As required by the HTTP protocol specification, this date is an HTTP full date sent in the RFC 1123-defined fixed length format. Example: `Sun, 08 Oct 1961 10:08:00 GMT`

The following HTTP response headers are provided in the response to all requests except those that result in a 204 (No Content) HTTP status code.

HTTP header name	Description
Content-Length	When used in a response, specifies the length of the response body. If omitted, the response does not contain a body.
Content-Type	When used in a response, specifies the MIME media type of the response body. This response header is provided any time the **Content-Type** header is provided and specifies a non-zero length.

Additional response headers

Some operations may return additional response headers beyond those described in "Standard response headers" on page 15. The following table describes these possible additional response headers. Operations that return these additional headers indicate that they do so in the operation description.

HTTP header name	Description
Location	The URI of the resources that was created by the operation. This header is provided for operations that complete successfully with an HTTP status code of 201 (Created).
X-API-Session	An opaque string that provides a cryptographically strong identifier of the API session that was created for the client. This header is provided in the response to a successful **Logon** operation.
X-Request-Id	A string that provides diagnostic information identifying the request from the perspective of the HMC. This same information is included in the API log entries that are recorded by the HMC for the request. If captured by the client from a response, a client application developer or support technician can use this information to locate the HMC API log entry corresponding to a particular request. The value of this header will be 64 characters or less. This header is provided in the responses to all requests.

Media types

The following media types are applicable to the use of the Web Services API, and thus may appear in the values of **Accept** or **Content-Type** header fields.

MIME media type	Description
`application/json`	JavaScript Object Notation (JSON), as described by RFC 4627. This media type is used by the Web Services API for both request and response representation for the majority of the operations in the API. The JSON text is encoded using the UTF-8 charset.
`application/vnd.ibm-z-zmanager-metrics`	Custom output format used for providing the results to the Get Metrics operation of the metrics service. The result text is encoded using the UTF-8 charset.
`application/xml`	Extensible Markup Language, used for the input and output formats for the Export Performance Policy and Import Performance Policy operations of the workload object. The XML text is encoded using the UTF-8 charset.
`application/octet-stream`	Binary data. This media type is used by the Web Services API for the request representation for the **Mount Virtual Media Image** operation of the virtual server object.

HTTP status codes

The HMC API provides standard HTTP status codes in the response to requests to indicate the success or failure of the request. Unless stated otherwise in the description of an operation, the following general interpretations of the status code values apply.

HTTP status code	Description/Causes
200 (OK)	The request has succeeded completely. A response body is provided that contains the results of the request.
201 (Created)	The request has succeeded completely and resulted in the creation of a new managed resource/object. The URI for the newly created managed resource is provided in a **Location** header. (POST methods only)
202 (Accepted)	The request was successfully validated and has been accepted to be carried out asynchronously.
204 (No Content)	The request succeeded completely, and no additional response information is provided.
400 (Bad Request)	The request was missing required input, had errors in the provided input, or included extraneous input. Additional information regarding the error is provided in an error response body that includes a reason code with additional information.
403 (Forbidden)	Multiple error conditions result in this status code: • The request requires authentication but no **X-API-Session** header was provided, or one was provided but the session ID was invalid. • The user under which the API request was authenticated is not authorized to perform the requested operation. • The ensemble is not operating at the management enablement level required to perform this operation.
404 (Not Found)	Multiple error conditions result in this status code: • The URI does not designate an extant resource, or designates a resource for which the API user does not have object-access permission. • The URI designates a resource or operation that is not supported by the HMC because it is currently the alternate HMC.
405 (Method Not Allowed)	The request specifies an HTTP method that is not valid for the designated URI.
406 (Not Acceptable)	The **Accept** header for the request does not include at least one content representation supported by the Web Services API.
409 (Conflict)	The managed resource is in an incorrect state (status) for performing the requested operation. Additional information regarding the error is provided in an error response body that includes a reason code with additional information.
413 (Request Entity Too Large)	The request includes a request body that is too large. Unless a different limit is specified for a particular operation, in general the largest request body accepted by the API is 64 KB.
415 (Unsupported Media Type)	The **Content-Type** header for the request specifies a representation that is not supported by the Web Services API.
500 (Server Error)	A server error occurred during processing of the request.
501 (Not Implemented)	The request specifies an HTTP method that is not recognized by the server (for any resource). **Note:** The response body that accompanies this error is not a JSON response body as defined in "Error response bodies" on page 18.
503 (Service Unavailable)	The request could not be carried out by the HMC due to some temporary condition.
505 (HTTP Version Not Supported)	The request specifies an HTTP protocol version that is not supported by the Web Services API.

Error response bodies

For most 4xx and 5xx HTTP error status codes, additional diagnostic information beyond the HTTP status code is provided in the response body for the request. This information is provided in the form of a JSON object containing the following fields:

Field name	Type	Description
http-status	Integer	HTTP status code for the request.
request-method	String	The HTTP method (DELETE, GET, POST, PUT) that caused this error response.
request-uri	String	The URI that caused this error response.
reason	Integer	Numeric reason code providing more details as to the nature of the error) than is provided by the HTTP status code itself. This reason code is treated as a sub-code of the HTTP status code and thus must be used in conjunction with the HTTP status code to determine the error condition. Standard reason codes that apply across the entire API are described in"Common request validation reason codes." Additional operation-specific reason codes may also be documented in the description of the specific API operations.
message	String	Message describing the error. This message is not currently localized.
stack	String	Internal HMC diagnostic information for the error. This field is supplied only on selected 5xx HTTP status codes.
error-details	Object	A nested object that provides additional operation-specific error information. This field is provided by selected operations, and the format of the nested object is as described by that operation.

Usage notes:

- The message provided in the **message** field is primarily intended as a convenience for use by developers when developing and testing client applications. Because it is not localized, it may not be appropriate for client applications to simply pass this message on to their clients when reporting errors to those upstream clients. Instead, client applications can use the value in the **reason** field as a key in obtaining a client-provided message that may be more appropriate to use.
- Because the reason code is treated as a sub-code of the HTTP status code, the same reason code value is often defined for multiple different HTTP status codes and has a different meaning in each case. For example, reason code 1 when considered for a 400 (Bad Request) status code has a different meaning than when considered for a 403 (Forbidden) status code. For this reason, client applications that make decisions based on the reason codes should always include checking the HTTP status code as part of the relevant logic (e.g.test for status code == 400 AND reason code == 1, not just reason code == 1 alone).

Common request validation reason codes

The Web Services API performs request validation on each request it receives to ensure the request is correctly formed and appropriate before it begins processing the request. Many errors of basic request syntax can occur on all or a large number of the operations provided by the API. Validation for these kinds of errors is done in a common way across all of the operations and results in a common (not request-specific) reason code being reported if errors are detected. Other validation operation-specific by nature, and results in operation-specific reason codes when errors are detected.

The following table provides the HTTP status codes and reason codes for common request validation. These status and reason codes may be reported on any of the operations of the API.

HTTP status code	Reason code	Description
400 (Bad Request)	1	The request included an unrecognized or unsupported query parameter.
	2	A required request header is missing or invalid.
	3	A required request body is missing.
	4	A request body was specified when not expected.
	5	A required request body field is missing.
	6	The request body contains an unrecognized field (i.e. one that is not listed as either required or optional in the specification for the request body format for the operation).
	7	The data type of a field in the request body is not as expected, or its value is not in the range permitted.
	8	The value of a field does not provide a unique value for the corresponding data model property as required.
	9	The request body is not a well-formed JSON document.
	10	An unrecognized X-* header field was specified.
	11	The length of the supplied request body does not match the value specified in the **Content-Length** header.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform the requested action.
	3	The ensemble is not operating at the management enablement level required to perform this operation.
	4	The request requires authentication but no X-API-Session-header was specified in the request.
	5	An X-API-Session header was provided but the session id specified in that header is not valid.
404 (Not Found)	1	The request URI does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.
	2	A URI in the request body does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.
	3	The request URI designates a resource or operation that is not available on the Alternate HMC.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.
	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	3	The operation cannot be performed because the object designated by the request URI is currently locked to prevent disruptive changes from being made.

Common request processing reason codes

Certain common error conditions can be encountered during the processing of many of the operations of the API. When they are encountered they are reported using the same HTTP status and reason code by any operation of the API that may encounter them.

These common request processing reason codes are listed in the following table:

HTTP status code	Reason code	Description
500 (Server Error)	19	An Asynchronous operation was terminated while running because the host HMC was restarted, or a failover to the alternate HMC occurred.
	Other in range 0 - 39	An internal processing error has occurred and no additional details are documented.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the SE is not currently communicating with an element of a zBX needed to perform the requested operation.
	3	This request would exceed the limit on the number of concurrent API requests allowed.

Use of chunked response encoding

For most API operations, the size of the response data is modest and therefore standard HTTP response payload transfer encoding is used. In this encoding, the length of the entire payload of the response message is provided in the response before any of the contents of the response payload are written to the socket connection. But some operations, such as the Get Inventory operation of the Inventory service and the Get Metrics operation of the Metrics service, can produce very large responses. Use of standard transfer encoding for these kinds of operations is inefficient for the HMC because it requires the entire response be generated and buffered before any of it is sent in order to compute and send the total length of the response body before sending any of the contents of the response data.

To avoid the need for the buffering the entire response, and to instead allow the response to be transmitted in smaller segments as they are prepared, operations that return large responses use HTTP chunked response encoding instead. Chunked transfer encoding is an HTTP V1.1 data transfer feature that allows the payload of the response message to be split into a sequence of smaller parts known as chunks, with the size of each chunk transmitted as part of the chunk rather than requiring the transmission of the size of the full response payload.

Chunked transfer encoding is defined in the HTTP/1.1 protocol standard, RFC 2616, cited earlier in this section.

The HTTP protocol standard requires that all HTTP/1.1 applications (client or server) be capable of receiving and handling chunked transfer-encoded messages, so the use of this encoding by the API HTTP server is within the options allowed by the protocol standard. However, since this format may be unexpected to naively-written applications, its use is limited by the API HTTP server to the special circumstances that warrant its use to improve performance or efficiency. Therefore, a client application can safely assume that an operation will not use chunked transfer encoding for its responses unless the use of this encoding is specifically mentioned in the description of the operation.

Filter query parameters

Some operations allow for the (optional) use of designated query parameters for conveying additional request parameters. Although query parameters can be used to convey various kinds of additional request information, most operations that make use of query parameters do so for the purpose of filtering the response entries to a subset of what would otherwise be returned. For example, this kind of filtering is typically provided on operations that are described as List operations (e.g. List Virtual Servers of Ensemble). This section describes the interpretation/handling of filter-type query parameters across all of the operations of the API.

As would be expected, if an operation is invoked without specifying any of its possible filter-type query parameters, the operation returns all of the result entries applicable to the request. For example, the List Virtual Servers of Ensemble operation invoked with no filtering query parameters returns all of the virtual server objects in the Ensemble to which the API user has access.

If one or more filter-type query parameters are specified, the combination of those parameters specifies a logical match expression that is evaluated against each entry that is a candidate for inclusion in the result to determine if the entry is included or not. Within that expression, there may be multiple occurrences of the same-named query parameter and/or there may be occurrences if differently-named query parameters. The query parameters are interpreted as a logical expression using the following rules:

- Multiple occurrences of the same-named query parameter are interpreted as a group that is connected by a logical OR operation among all of query parameters with the same name. An entry remains a candidate for inclusion in the result as long as it matches at least one of the values specified for this particular query parameter.
- Occurrences of differently-named query parameters are first organized into OR'ed groups as mentioned above, and then these groups are interpreted as being connected by logical AND operations. Thus an entry is included in the result only if it matches at least one value from each of the differently-named groups of parameters.
- As an example, a query string of “name=fee&type=fie&name=foe&type=fum” is interpreted as specifying the expression (name=fee OR name=foe) AND (type=fie OR type=fum). Note that the order in which the query parameters appear in the string is not important.

As a filter-type query parameter is applied against a candidate entry, it is determined to match or not as follows:

- If the query parameter is of data type String, the parameter's value is interpreted as a regular expression pattern and is considered to match if the corresponding String property of the candidate entry matches the pattern.
- If the query parameter is of data type String Enum, the parameter's value is compared against the corresponding Enum property of the candidate entry and is considered to match if they are exactly the same value.

Regular expression syntax

The values of String-type filtering query parameters are interpreted as regular expressions. The regular expression syntax used is the same as that used by the Java programming language, as specified for the java.util.regex.Pattern class. Documentation on that syntax can be found at on the following web page: http://download.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html

Chapter 4. Asynchronous notification

The Web Services API includes an asynchronous notification facility by which client applications may subscribe to and receive notification messages regarding a set of predefined management events. These events include:

- Addition and removal of managed objects to/from the inventory of resources that are managed by the HMC.
- Changes to specified properties of managed object instances.
- Changes to the operational status of managed objects.
- Completion of asynchronously processed jobs.

The zManager notification facility is based on the Java Message Service (JMS) architecture and API for exchanging messages among two or more applications.

JMS basics

In the JMS model, message-based communication between producing and consuming applications is coordinated by an intermediate component known as a message broker that acts as the clearinghouse for message exchange. The message broker provides as a registry of the available destinations to which messages can be posted, and a store for messages that have been posted but not yet consumed.

Applications acting in the role of message producer create messages and post them (via the broker) to the destination appropriate for the type of message. The messages are associated with the destination and retained by the broker until they have been consumed by interested applications.

Applications acting in the role of message consumer connect to a message destination (again, via the broker) in order to express interest in receiving messages posted to it. As messages are posted to the destination by producers, the broker makes the messages available to interested consumers which then receive and process the message.

In the Web Services API notification facility, the HMC acts both as the message broker and the message producer for API notification messages. API client applications act as message consumers.

For the broker function, the HMC includes an integrated JMS message broker implementation based on Apache ActiveMQ, an open, standards-based implementation of JMS. This integrated broker is configured to allow internal HMC function to act as message producers, and to allow external applications to connect as message consumers. However, external applications cannot produce and send message using the HMC integrated broker.

Connecting to the API message broker

As part of the Web Services API, the HMC provides an integrated JMS message broker based on Apache ActiveMQ Version 5.2.0. This message broker is active on the HMC whenever the Web Services API is enabled.

When active, the integrated broker listens for client connections using the following transports supported by ActiveMQ:

- OpenWire flowing over SSL connections, listening port: 61617.
- STOMP (Streaming Text Oriented Messaging Protocol) flowing over SSL connections, listening port 61612.

The broker is enabled for the SSL version 3 and TLS version 1 protocols on these SSL ports.

The listening ports for the message broker are fixed port numbers and are not subject to customer reconfiguration. Thus, client applications can treat them as well-known port numbers rather than requiring customer input when configuring the networking parameters the client will use to connect to the HMC.

In order to connect to the integrated message broker, clients must provide a valid HMC user name and password in order to identify the HMC user making the connection. This information is validated using the standard HMC user authentication mechanisms before allowing the connection to succeed. The integrated message broker does not allow any anonymous or unauthenticated connections.

Per-session notification topics

As part of its access control enforcement, the Web Services API limits the distribution of notification messages to clients that have object-access permission to the managed object for which the notification was generated.

In order to accomplish this, the API does not define a single (or fixed number of) notification topic to which all messages are posted and from which any or all client can receive message. Rather, the API uses per-session notification topics.

In this approach, each API session is associated with two JMS destinations that are created by the HMC when the session is created and are used for providing notifications destined to that session. The names of these destinations are returned as part of the results from the Login operation. Each session has the following per-session notification destinations:

- An object notification topic, used by the HMC to send notifications that pertain to the inventory and status of managed objects that this session has permission to access.
- A job notification topic, used by the HMC to send notifications that pertain to the status of asynchronous operations that are initiated by this session.

The session is also associated with an HMC user (identified during API session login) that in turn has a set of object access permissions defined for it that determine the managed resources that it is authorized to access.

As notifications messages are created for managed resource changes or job completions, they are distributed to sessions according to the object access permissions of those sessions. More specifically, when a notification is generated pertaining to some managed resource, it is published to the object notification topics of all sessions for which the related API user has object-access permission to that managed resource, and is omitted from the object notification topics of sessions for which the user does not have object-access permission. As a result, a particular API session will have access to all notifications for all managed resources to which its API user has access permission, but will not have access to notifications for managed resources that it does not.

Notification messages for job completion are sent only to the job notification topic of the API session that initiated the asynchronous processing represented by the job.

Notification message formats

Four types of notification messages are provided by the API: property change notifications, status change notifications, inventory change notifications, and job completion notifications.

The JMS messages created for all types of notifications share a common set of message characteristics and header fields, which are extended with additional header fields and message body formats that vary by the type of notification.

Common message characteristics

The characteristics described in this section apply to all notification messages published by the Web Services API.

Message format

The following JMS message characteristics apply to all notification messages sent by the Web Services API. These characteristics are echoed in the message themselves in the values of the standard JMS message header fields.

Characteristic	Description
Destination	The per-session object or job notification topic as indicated below for each type of notification.
Message type	Text message. The contents of the body vary by the type of notification.
Delivery mode	Nonpersistent.
Expiration	No expiration.
Priority	4 (highest normal priority)
Message ID	A unique message ID for the message.
Correlation ID	Not set by the API.
Timestamp	The time, from the HMC's perspective, that this message was sent.

In addition to the standard JMS message headers, the following additional message properties are provided in all notification messages to allow for message selection:

Message Property Name	Description
notification-type	The type of notification contained in this message, varies by notification type.
session-sequence-nr	The sequence number of this notification within the session. This number starts at 0 when the API session is created, and is incremented each time a notification message is published to this session.
global-sequence-nr	The sequence number of this notification from the HMC. This number starts at 0 when the HMC is started, and is incremented each time a notification message is published to any API session.
object-uri	The current value of the **object-uri** property (canonical URI) of the managed object for which the notification is being emitted.
object-id	The current value of the **object-id** property (durable unique identifier) of the managed object.
element-uri	The current value of the **element-uri** property of the element object for which the notification is being emitted. This message property is included only when the message pertains to an element object of a managed object.
element-id	The current value of the **element-id** property (local identifier) of the element object. This message property is included only when the message pertains to an element object of a managed object.
class	The current value of the **class** property (kind of object) of the managed object, i.e. the kind of object for which the notification is being emitted.
name	The current value of the **name** property (display name) for the object for which the notification pertains. **Note:** Note: In some circumstances the **name** property may be unavailable, in which case this field is set to an empty string. This may occur, for example, if a property change occurs and is to be reported on very shortly before (essentially concurrently) with the removal of that object from the inventory.

When a notification message pertains to an element object, the message includes **element-uri** and **element-id** fields in addition to **object-uri** and **object-id** fields. The element-* fields identify the element

object instance while the object-* fields identify the containing managed object instance. In this case, the **class** field provides the class of the element object, and the **name** field provides the name of the element object.

When a notification message pertains to a managed object, the message contains **object-uri** and **object-id** fields but not the element-* fields and the **class** field provides the class of the managed object and the **name** field provides the name of the managed object.

Grouping of notifications

A particular managed resource may often experience a series of changes that occur in rapid succession. This might occur, for example, when a user uses an object's Details task in the HMC UI to make a set of changes to the object's properties and then selects the Finish button to complete the task. All of the pending property changes collected by the task will be made on the managed object in very quick succession in response to the Finish button is selected, rather than before as the user was interacting with the task.

In order to reduce the notification traffic in these situations, the notification messages have been designed to allow the HMC to report multiple changes of the same type (e.g. property changes, status changes) for the same managed resource in a single message rather than generating a distinct message for each change. In order to do such grouping, the HMC may delay generation of a notification message for a change for a brief period of time in order to allow coalescing of that change report with others that occur for the same managed resource within the coalescing time window. This optimization will be performed while maintaining the following characteristics:

- Grouping of notifications may be done for property change and status change notifications, but will not be done for other notification types.
- Notifications will be buffered for a maximum of 1 second.
- The grouping of change reports will not obscure a client's ability to correctly observe the temporal ordering of the individual changes made to a particular object or between objects based on the messages sent to a session. That is, notification messages will always be generated so that the ordering of the messages as determined by their session sequence numbers reflects the temporal order in which the changes were actually made. All of the changes reported to a session in a message with a lower session sequence number will have occurred before any of the changes reported in a message with a higher session sequence number. Further, the ordering of change reports within a particular message reflects the order in which they occurred to that object as well.
- Coalescing of multiple changes into a single notification message will occur for at most a single pending notification message (thus, of a single type) at a time. If a need arises to report a change of a different type than is currently pending (for example, a need to report a status change when there is currently a set of pending property change reports), coalescing will end for that pending message and it will be posted to notification topics as appropriate. This is necessary in order to maintain the API client's ability to correctly observe temporal ordering.
- The grouping of change reports into notification messages occurs independently for each session, so that one session may receive a different distribution of change reports across notification messages than another session.

The degree to which message grouping occurs or not depends on the timing of changes and possibly other considerations and thus is to be strictly considered an optimization and not guaranteed behavior. Client applications should infer no particular semantic significance to change reports being delivered in a single message vs. a sequence of messages.

Status change notification

A Status Change notification is emitted by the API to report changes to the **status** property of a managed object.

Characteristic	Description
Destination	The per-session object notification topic for each API session that is authorized to receive the notification.

In addition to the common JMS message headers described above, the following additional message properties are provided for this type of notification:

Message property name	Description
notification-type	Contains the value **"status-change"**.

The body of a Status Change notification message is a JSON representation of an object that contains the following fields and values:

Field name	Type	Description
change-reports	Array of objects	An array of nested change-report objects, the format of which is described in the next table. The order in which these objects appear in this array reflects the temporal order in which the changes occurred.

Each nested change-report object has the following fields and values:

Field name	Type	Description
old-status	String	The old (previous) value of the **status** property for the object. The value of this field will be one of the possible enumeration values for the **status** property as defined for this class of object.
old-additional-status	String	The old (previous) value of the **additional-status** property for the object. The value of this field will be one of the possible enumeration values for the **additional-status** property as defined for this class of object.
new-status	String	The new (current) value of the **status** property for the object. The value of this field will be one of the possible enumeration values for the **status** property as defined for this class of object.
new-additional-status	String	The new (current) value of the **additional-status** property for the object. The value of this field will be one of the possible enumeration values for the **additional-status** property as defined for this class of object.
has-unacceptable-status	Boolean	The value of the **has-unacceptable-status** property of the object, based on its new status. If true, the object is now considered to have unacceptable status because its current status is not one of the configured acceptable status values for this object.

Property change notification

A Property Change notification is emitted by the API to report changes to the properties of a managed object where the data model description indicates that modification notification support (qualifier “pc”) is available for that property.

Characteristic	Description
Destination	The per-session object notification topic for each API session that is authorized to receive the notification.

In addition to the common JMS message headers described above, the following additional message properties are provided to allow for message selection:

Message property name	Description
notification-type	Contains the value **"property-change"**.
property-names	Blank-separated list of the names of the properties for which change reports are provided in body of this notification message. The list always includes a leading and trailing blank so that a property name can be specified as a blank-delimited word in a message selector to avoid matching unintended properties that have the desired property name as a substring.

The body of a Property Change notification message is a JSON representation of an object that contains the following fields and values:

Field name	Type	Description
change-reports	Array of objects	An array of nested change-report objects, the format of which is described in the next table. The order in which these objects appear in this array reflects the temporal order in which the changes occurred.

Each nested change-report object has the following fields and values:

Field name	Type	Description
property-name	String	The name of the property (as specified in the object's data model) that has changed.
old-value	Based on model	If the property is not a container-type property, this field contains the old (previous) value of the property for the object. The value of this field will be of the data type indicated for this property in the object's data model. If the property is a container-type property (i.e. marked with the (c) qualifier), this field does not provide the complete previous value. Rather, it provides an array of entries that have been removed from the value of the container property. The value of these entries will be of the data type indicated for the property in the object's data model. If no entries have been removed, null is provided.
new-value	Based on model	If the property is not a container-type property, this field contains the new (current) value of the property for the object. The value of this field will be of the data type indicated for this property in the object's data model. If the property is a container-type property (i.e. marked with the (c) qualifier), this field does not provide the complete new value. Rather, it provides an array of entries that have been added to the value of the container property. The value of these entries will be of the data type indicated for the property in the object's data model. If no entries have been added, null is provided.

Inventory change notification

An Inventory Change notification is emitted by the API to report the addition or removal of a managed object to/from the current inventory of resources that are being managed by zManager. This occurs when managed resources are created or deleted, but also may occur in other situations, such as when zManager reestablishes its inventory of (already-existing) managed resources upon restart of the primary HMC.

For some kinds of managed objects, an Inventory Change notification is also be emitted by the API to report the addition or removal of an element of a managed object. Such notifications do not occur for all elements, but rather only when specifically described in the documentation for a class of managed object.

Because an Inventory Change notification may be generated more than once for the same conceptual object, these notifications cannot be interpreted as designating a resource creation action.

Characteristic	Description
Destination	The per-session object notification topic for each API session that is authorized to receive the notification.

In addition to the common JMS message headers described above, the following additional message properties are provided to allow for message selection:

Message property name	Description
notification-type	Contains the value **"inventory-change"**.
name	Not provided for this notification. Always an empty string.
action	The value **"add"** when the object has been added to the inventory, or **"remove"** when it is being removed.

The body of an inventory change notification is null.

Job completion notification

A Job Completion notification is emitted by the API to report that an operation that runs asynchronously to the client application has completed its processing.

Asynchronous operations are those that response with an HTTP status code of 202 (Accepted) when requested by the client. A Job Completion Notification message is sent to the API session that initiated the job when such an operation completes, and provides to the client application the URI of the job that has completed so the client application can use the Query Job Status operation to obtain results for the job.

Characteristic	Description
Destination	The per-session object notification topic for each API session that is authorized to receive the notification.

In addition to the common JMS message headers described above, the following additional message properties are provided to allow for message selection:

Message property name	Description
notification-type	The value **"job-completion"**.
job-uri	The URI identifying the asynchronous job that has just completed execution.

The body of a job completion notification is null.

Chapter 5. Data model definitions

This chapter covers data model concepts and shared data model schema elements.

Data model concepts

zManager provides resource management and control functions for the various resources that comprise a System z ensemble environment. In performing these functions, zManager establishes a separation between those aspects of resource management that are handled entirely by system firmware, and the other aspects for which customer or installation visibility, configuration and control is appropriate.

In order to specify the external aspects in a succinct way, the web Services API is described in this document in terms of a conceptual data model that it offers for the resources that it manages. This data model is an information structuring technique that conceptually defines the kinds of resources that are managed by zManager and for each, the information that is available for and the operations that can be performed on resources of that kind. This data model is intended to provide the complete perspective that clients of the API can have regarding the logical resources of the system while insulating them from implementation details.

Objects in the data model

The manageable resources of the environment are represented in the management system as entities referred to as objects. Each distinct manageable resource is represented by a separate object instance, and the life cycle of an instance corresponds with the lifecycle of the manageable resource it represents. For example, for physical entities, such as an IBM blade in a zBX, the object that represents it are created implicitly when the physical entity is attached to and configured to be (or entitled to be) part of the system. This object continues to exist so long as the IBM blade remains entitled on the system. For some logical entities, such as the virtualization management functions on an IBM blade, the object that represents it (virtualization host) is also implicitly rather than explicitly created. For other logical entities, such as a virtual server on an IBM blade, the creation of the management model object instance for it is, in fact, the mechanism that triggers the creation of the corresponding manageable resource in the system.

There are different kinds of manageable physical or logical resources in the system, and each kind manifests different observable characteristics. As a result, there are different classes of objects in the data model. Objects of the same class represent the same kind of resource and provide a defined set of *properties* that capture the attributes of that kind of resource that the Web Services API exposes.

Managed objects and element objects

The object classes defined in the data model fall into one of two main categories: managed object classes (or simply managed objects), and element object classes (element objects).

These two categories are very similar in that they are both, ultimately, unordered collections of named properties that capture the key attributes of a resource instance. The categories differ primarily in how prominently they are handled in the API: the way that instances of them are designated to perform operations on them, and the degree to which API facilities such as inventory and change notification can be offered for objects in that category.

Managed objects are the first-class entities in the data model and the API. They represent the primary manageable resources of the system, such as ensembles, blades, virtual servers and workloads. These kinds of objects typically appear prominently in the main displays of the HMC user interface.

Instances of managed objects are registered and indexed in the zManager managed object registry, and thus can be directly referenced by URIs that form a relatively "flat" namespace. The URI of a managed

object designates its object instance based on its class and a unique, durable, UUID-based identifier called an object ID. For example, the URI of a virtual server is of the form `/api/virtual-servers/{vs-object-id}` where the identifier at the end of the URI is globally unique. Inventory change, property change, and status change notifications can be generated for managed objects.

In comparison, element objects represent the secondary or more-detailed aspects of the system. Examples include the virtual network adapters of a virtual server, or the performance policies of a workload. These kinds of entities do not generally appear in the main displays of the HMC user interface, but rather are displayed only within particular management tasks offered by the UI.

Instances of element objects are not directly registered in the zManager object registry, but rather are associated with or "attached to" some containing or related managed object instance. As a result, access to these elements is indirect, through the containing managed object. The URIs that designate element objects are hierarchical in nature, with the leftmost part of the URI identifying the managed object to which the element is attached. For example, the URI for a virtual disk of a virtual server is of the form `/api/virtual-servers/{vs-object-id}/virtual-disks/{disk-id}` in which the *{disk-id}* at the end is only necessarily unique within the context of the related virtual server. Inventory, property and status change notifications are not provided for element objects. Instead, when changes to elements are reported, those changes are done via property change notifications emitted for the associated managed object.

Properties in the data model

Object in the management system contain, fundamentally, unordered collections of name/value pairs called properties that capture the key characteristics of the manageable resources they represent. The defined set of named properties that are maintained for a particular kind of resource constitutes the specification of the data model class for that kind of resource.

As a result, in the chapters that follow, the description of the management interfaces for a class of resource begins with a Data Model section that specifies the properties that are exposed by the API for that kind of resource.

Each property has a name, a data type, and a semantic description in prose.

The property name is the programmatic identifier of the property. This identifier is used within requests and responses to indicate that a field represents a particular property of the data model. It is the "name" part of the name/value pair that is the property.

The property data type indicates the kind of information that can be represented by the property, just as a variable's data type indicates the kind of information that can be stored in a variable. The data type provides information on the nature of the "value" part of the name/value pair that is the property.

Property characteristics

Properties are classified as being either writeable or read-only from the perspective of an API client application.

Writeable properties are ones that can have their values read by Get <class> Properties or similar operations and can also have their values directly changed by Update <class> Properties operations. Properties that are classified as read-only can have their values read by using Get <class> Properties or similar operations, but cannot have their values changed directly.

Although properties that are classified as read-only cannot have their values changed directly, their values may nonetheless be affected by other operations supported by a class of object. For example, a class of object might include an **is-enabled** property that is classified as read-only because the enabled state of the resource cannot be affected by a simple Update <class> Properties operation on that **is-enabled** property. However, this object might also define a Change State or Enable operation that can perform this enabling, and as a side effect will alter the value of the **is-enabled** property.

In addition to the read-only vs. writeable classification, properties defined for managed object also can differ in whether changes to them result in property change or status change notifications being emitted for the managed or not. For properties that have property or status change support, these notifications are emitted asynchronously by the API any time the value of the property changes, whether that change was made via this API, the HMC UI, or implicitly by the system. Changes to the values of properties for which change notification support is not provided do not result in such notifications.

In the tables of properties that appear within this document, the characteristics of properties are indicated by qualifier annotations in parenthesis following the property name. The qualifiers have the following meanings:

Qualifier notation	Description
(w)	The property is a writeable property. Any property that lacks this qualifier is considered read-only and thus is not directly modifiable.
(ro)	Although this property is writable when present in other managed object classes, it is read only in this class. This qualifier is only used when a managed object class specializes the definition of a Base Managed Object Property and overrides the writeable characteristic of the base definition.
(pc)	Change to this property's value will result in Property Change notifications.
(c)	The property is a container-type property for which deltas (changes) are reported in Property Change notifications rather than complete old and new values.
(sc)	Change to this property will result in Status Change notifications.
(mg)	This property represents a performance or utilization metric of the object that is included in a metric group available via the Metrics Service of this API. The value of this property may change very frequently and, therefore, property change notifications are not emitted for changes to this property. Client applications interested in obtaining metric information frequently should obtain this information through use of the Metrics Service of this API.

Shared data model schema elements

The data-model schema fragments in this section define groups of properties that are used in common ways in specifying the data models for the managed object classes defined in the API.

The description of the data model for a specific object class specifies the shared schema elements it is incorporating within the Data Model section of that description, if any. It will also include a description of the specializations that apply to that class's use of the shared schema, such as additional constraints on properties, class-specific values for properties, etc.

Base managed object properties schema

This data-model fragment contains the basic properties that are present in the representation of many of the managed object types that represent manageable resources.

Name	Qualifier	Type	Description
object-uri	—	String (1-255)	The canonical URI path that designates this managed object instance and serves as the primary reference and retrieval key for this instance. The URI path is formed based on a unique and permanently-assigned object ID (see the **object-id** property in the next row of this table), and as result, an object's URI path will not change as a result of changes to properties of the object. Further, this canonical URI path is independent of the containment hierarchy and thus will not change if this object instance is moved within the hierarchy.

Name	Qualifier	Type	Description
object-id	—	String (36)	The object identifier for the managed object instance. This value is unique in space and time, and is permanently associated with this instance while it is managed by this HMC or its associated alternate HMC. (If the instance is removed from this HMC and later managed by another HMC, it will have a new and different object identifier when managed by that other HMC.) It is generated by zManager and assigned when the managed resource is created or first discovered, and is immutably thereafter. As example, a managed object's object ID will not change as a result of changes to display name, changes in the location of this resource in the containment hierarchy, or across restarts of the HMC.
parent	—	String (1-255)	The canonical URI path of the managed object that is conceptually the parent of this object in the containment hierarchy. This property is null for objects that do not have a parent.
class	—	String (1-32)	The class of resource represented by this managed object. Each distinct class of resource has a different type name, while all instances of the same type share the same type name. The specific value used for a class of object is specified in the Data Model section for that object type. Example: **"virtual-server"**.
name	(w)(pc)	String (1-64)	The current display name of the managed object as defaulted or specified for the object. This is the simple name of this object, not qualified by containment hierarchy. This name is required to be non-blank and non-null. Some resource types do not support the setting of a user-assigned display name. For such objects, this property is not settable, and instead always provides a name assigned by zManager.
description	(w)(pc)	String (0-1024)	Arbitrary text providing additional descriptive information about this managed resource. This information is retained for the resource and may be shown as part of the object's details on the user interface, but is otherwise not generally used by zManager. This property may be null.
is-locked	(pc)	Boolean	The object is locked and thus disruptive actions or tasks cannot be performed on it.

Operational status properties

Many (but not all) classes of managed objects support the concept of operational status. That is they maintain information about the current functional state (Not Communicating, Not Operating, etc.) of the managed resource and whether that current functional state is considered acceptable (not alert causing) or not. If a class of object supports the operational status concept, it provides the standard properties defined in the following table (referred to as the operational status properties) in addition to those defined earlier in this section.

Unless stated to the contrary, any object class data model that includes the Base Managed Object Properties schema should be understood to also provide these operational status properties as well. For object classes for which that is not the case, the data mode description will specifically point out that operational status and thus these operational-status-related properties are not provided for that object.

The operational status properties are as follows:

Name	Qualifier	Type	Description
status	(sc)	String Enum	The current operational status of the managed resource. The possible status values vary by managed object class and are specified in the description of each managed object class that provides this property.
additional-status	(sc)	String Enum	A qualifier to the **status** property used by selected object classes to provide finer grained operational status tracking.

Name	Qualifier	Type	Description
acceptable-status	(w)(pc)	Array of String Enum	The set of operational status values that the managed resource can be in and be considered to be in an acceptable (not alert causing) state.
has-unacceptable-status	(sc)	Boolean	If true, the current operational status of the managed resource is not one of the acceptable statuses for this resource.

Chapter 6. General API services

This chapter describes the services that are provided by the Web Services API for creating and deleting API sessions and performing other general functions.

General API services operations summary

Table 6. General API services: operations summary

Operation name	HTTP method and URI path
"Query API Version" on page 38	`GET /api/version`
"Logon" on page 39	`POST /api/sessions`
"Logoff" on page 42	`DELETE /api/sessions/this-session`
"Query Job Status" on page 44	`GET /api/jobs/{job-id}`
"Delete Completed Job Status" on page 46	`DELETE /api/jobs/{job-id}`

Table 7. General API services: URI variables

Variable	Description
{job-id}	The identifier of an asynchronous job associated with this user, as returned in the response of the operation that initiated the job.

Session management services

Almost all operations of the Web Services API are requested and carried out in the context of an API session that is used for determining the client's authority to access managed resources and perform requested operations, and is also used to scope the delivery of asynchronous notifications. An API session is an HMC concept that is independent of and layers on top of network-related considerations such as a TCP/IP socket connection. As a result, a single API session may span multiple TCP/IP socket connect/disconnect sequences from the same client.

Sessions are created upon request from a client by using the **Logon** operation, and may be explicitly terminated by a client using the **Logoff** operation. Sessions may also be terminated by the HMC due to inactivity when no requests are made using the session over a period of 6 hours. (This session timeout is not configurable.) However, termination of a session due to inactivity will not occur as long as a client application uses the API's notification facility to maintain a JMS subscription to one or more of the session's JMS notification topics. The existence of such a subscription is considered by the HMC to indicate that a client is still using the session and thus it is not terminated even if no requests are made using it.

The scope of a session is the particular HMC instance on which it was created via a **Logon** operation. That is, an API session created on one HMC of a primary/alternate HMC pair is not also available on the alternate HMC of that pair. Nor will that session (and its associated **session-id**) be available on the other HMC should a primary/alternate role switch occur due to a failure of the primary HMC. After a primary/alternate role switch, client applications will need to establish new sessions by making Logon requests again.

Sessions are identified by clients using a **session-id**, which is a string of up to 64 characters in length that is returned to the client in the results from a successful **Logon** operation. This string is generated in a cryptographically-secure manner. A **session-id** string is a form of authentication credentials for a user equivalent in power to a user's user ID and password. Because of this, a **session-id** should be transmitted only within SSL connections.

In order to indicate that subsequent requests are to be performed in the context of a designated session, the client supplies the appropriate **session-id** to the HMC in each such subsequent request. This is done by supplying the **session-id** as the value of the **X-API-Session** HTTP header which is an application-specific header defined by and recognized by the HMC.

The **Logon** and **Query API Version** operations are the only two operations in the Web Services API that can be performed without an API session so requests for these operations do not need to provide the **X-API-Session** HTTP header. All other operations are valid only in the context of an API session and thus requests for all other operations must supply an **X-API-Session** header with a valid **session-id** in order to be successfully executed.

Query API Version

The **Query API Version** operation returns information about the level of Web Services API supported by the HMC.

HTTP method and URI

```
GET /api/version
```

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`api-major-version`	Integer	The major-version part of the API version in effect for this session
`api-minor-version`	Integer	The minor-version part of the API version in effect for this session
`hmc-version`	String (5-8)	The version number of the HMC firmware. This is a string of the form *v.r.m*, where each of *v*, *r* and *m* can be one or two digits. Example: "2.11.1".
`hmc-name`	String (1-16)	The name assigned to the HMC

Description

This operation returns name and version information for the HMC and the HMC API.

This operation can be requested without an API session being open, i.e. no **X-API-Session** header, and **session-id** is required on input.

This operation can be invoked on the alternate HMC.

For more information about the function included in each API version, see “Summary of API version updates” on page 5.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in“Response body contents.”

Under normal conditions, no error response codes are returned by this request. (HTTP Status code 500 could possibly result if internal HMC errors occur.)

Example HTTP interaction

Request:

```
GET /api/version HTTP/1.1
```

Response:

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 20 Jul 2011 17:22:23 GMT
content-type: application/json;charset=UTF-8
content-length: 119
{
   "api-major-version": 1,
   "api-minor-version": 1,
   "hmc-name": "HMCR32PRI",
   "hmc-version": "2.11.1"
}
```

Logon

The **Logon** operation establishes an API session with the Web Services API.

HTTP method and URI

POST /api/sessions

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
userid	String	Required	The name of the HMC user to be associated with the new API session. This name may be of arbitrary length, i.e. the HMC does not have a defined maximum length.
password	String	Required	The password used to authenticate the HMC user identified by the **userid** field. The required length and valid characters are determined by the password policy in effect for the user ID.
new-password	String	Optional	A new password to be established for the user defined by the **userid** field. The required length and valid characters are determined by the password policy in effect for the user ID.

The largest request body accepted by this operation is 512 bytes. Requests with bodies that exceed this maximum are rejected with an HTTP status 413 (Request Entity Too Large) response.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`api-session`	String (1-64)	The **session-id** of the newly created session. The client must specify this value in the **X-API-Session** header of all subsequent requests that are to be performed in the session.
`notification-topic`	String (1-128)	The name of the JMS topic the HMC will use to send object-related notification messages to this session.
`job-notification-topic`	String (1-128)	The name of the JMS topic the HMC will use to send job-related notification messages to this session.
`api-major-version`	Integer	The major-version part of the API version in effect for this session
`api-minor-version`	Integer	The minor-version part of the API version in effect for this session
`password-expires`	Integer	The time interval, in days, until the user's current password expires. A value of 0 indicates that the password will expire within the next 24 hours. A value of -1 indicates that the HMC does not enforce password expiration for this user, however, if this user is authenticated with an external authentication mechanism (e.g. LDAP) such expiration might be enforced by that mechanism.

Description

This operation opens a new API session with the Web Services API. Authentication is performed as part of this process.

The characteristics and permissions of an HMC user are specified in an HMC User Profile or User Template. The user name provided in the **userid** field of the request body is used to select a corresponding User Profile/Template based on the name. If such a User Profile/Template is found, the client's authority to operate as this HMC user is authenticated by validating the password provided in the **password** field using the authentication method specified in the User Profile/Template. If the password authentication is successful, a new API session is created and the session-id for the new session is provided in the **api-session** field in the response from this operation. This same value is also provided by an **X-API-Session** HTTP header field in the response.

If the request specifies an **X-API-Session** HTTP header field on input (indicating that this operation be performed under some designed session), the logon request fails and status code 400 (Bad Request) is returned.

If an HMC User Profile/Template corresponding to the user ID field does not exist, or if the password validation fails, the logon request fails and status code 403 (Forbidden) is returned. There is no reason code to distinguish these reasons for the failure. If the User Profile/Template is marked as disabled or the associated password has expired, or if the or if the User Profile/Template is not configured to allow use of the API, the login request also fails with status code 403 (Forbidden) and a reason code identifying the specific cause.

If user authentication is successful and the request body contains the optional **new-password** field, the password associated with the User Profile/Template is changed to the specified new value as part of the **Logon** operation. If the new password does not meet the requirements of the password policy in effect for this User Profile/Template or if the User Profile/Template password is not changeable because it is managed by an external authentication mechanism, the request fails with status code 400 (Bad Request) and a reason code indicating the cause of the failure.

As part of establishing the new API session, JMS topics are created that will be used by the HMC to send object-related and job-related notification messages to this session and the names of these topics are provided in fields of the response body. The name of the topic used for object-related notifications is provided in the **notification-topic** field of the response, and the name of the topic used for job-related notifications is provided in the **job-notification-topic** field.

This operation can be invoked on the alternate HMC, however password changes cannot be requested (i.e. the **new-password** field cannot be provided) in this case.

Authorization requirements

This operation has the following authorization requirement:

- The HMC User Profile or User Template selected by the **userid** field must be configured to allow use of the Web Services API.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 40.

The following HTTP status codes are returned for the indicated operation-specific errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	12	The request specified an **X-API-Session** header, which is interpreted as an attempt to unnecessarily logon again when already logged on.
	42	Password changes cannot be requested when logging on to the alternate HMC.
	43	The password for this user cannot be changed, for example because it is managed in an external authentication mechanism such as LDAP.
	44	The new password does not conform to the requirements of the password policy in effect for this user.
	45	The user's password has expired and no new-password field was specified.
403 (Forbidden)	0	User authentication failed.
	40	The user is disabled.
	41	The user is not authorized to use the HMC Web Services interface.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

The **Logon** operation checks for and prevents requests that specify an **X-API-Session** header on input in order to detect client applications that unnecessarily log on again when already logged on. It is valid to have multiple sessions, but in order to more explicitly indicate that this is desired, the client application needs to request each logon without referencing any existing session.

Example HTTP interaction

```
POST /api/session HTTP/1.1
content-type: application/json
content-length: 58
{
   "password": "12345678",
   "userid": "APIUSER"
}
```

Figure 1. Logon: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 02 Nov 2011 18:41:27 GMT
x-api-session: 4hy7c4nogldz4b59ajegzb1dulec641ziyv6uf73zs43205edv
content-type: application/json;charset=UTF-8
content-length: 207
{
   "api-major-version": 1,
   "api-minor-version": 1,
   "api-session": "4hy7c4nogldz4b59ajegzb1dulec641ziyv6uf73zs43205edv",
   "job-notification-topic": "APIUSER.229job",
   "notification-topic": "APIUSER.229",
   "password-expires": 29
}
```

Figure 2. Logon: Response

Logoff

The **Logoff** operation closes an API session with the Web Services API.

HTTP method and URI

DELETE /api/sessions/this-session

Description

This operation closes an API session with the Web Services API.

The session to be closed is indicated by the **session-id** in the **X-API-Session** header of the request. If the **session-id** designates an open session, the API session is closed and status code 204 (No Content) is returned. Closing of the API session includes closing/deleting any Metrics Service retrieval contexts or JMS notification topics associated with the session. However, asynchronous actions initiated by the session continue to run.

Once a session is closed, its **session-id** is no longer valid for use in subsequent Web Services API requests. Attempts to do so will result in the same errors as any other attempt to use a session-requiring operation without providing a valid **session-id**.

This operation can be invoked on the alternate HMC.

Authorization requirements

This operation has the following authorization requirement:

- No explicit authorization is required, however the client application must possess and present a valid **session-id** of the session to be closed.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned with no response body.

The following HTTP status codes are returned for the indicated operation-specific errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/session/this-session HTTP/1.1
x-api-session: zkspmapxgtcasy5uixmtwuaudqe8ha6fy0006bzmxsm2bd8yo
```

Figure 3. Logoff: Request

```
204 No Content
date: Wed, 20 Jul 2011 18:33:56 GMT
x-request-id: Sx32 Rx0
server: zSeries management console API web server / 1.0
cache-control: no-cache
```

Figure 4. Logoff: Response

Asynchronous job processing

Some of the operations provided in the Web Services API may take a significant amount of elapsed time to complete. In order to optimize the usage of HMC session resources and to allow the client application the opportunity to perform other processing, such long-running operations are structured to be executed asynchronously (rather than synchronously) from the perspective of the client application.

In a synchronous operation, the Web Services API does not respond to the client application's request until all of the processing associated with the request is complete (successfully or in error) and the API can provide a final result status for the operation. The client application thread is typically blocked (not running) during this time.

By contrast, in a function that operates asynchronously, the Web Services API performs just the minimal front-end validation and set up work needed to accept the request to perform the indicated operation, and then quickly returns an HTTP 202 (Accepted) result to the client indicating that the operation request has been started but is not yet finished. Along with the HTTP 202 (Accepted) result, the client application is provided with a URI that represents the asynchronous job that is in progress. This URI is of the form `/api/job/{job-id}`.

At any point after receiving the HTTP 202 (Accepted) result, the client application can invoke the **Query Job Status** operation described in this section to determine if the job is complete or not. If the job is not yet complete, the **Query Job Status** request returns an indication that the job is still running. If the job is complete, the **Query Job Status** request returns an indication that the job is now complete along with the final status code, reason code and result data associated with the now-completed asynchronous processing. Once complete, job status is retained by the HMC for a minimum of 4 hours to allow the client application time to retrieve the results, but this status and results are not held indefinitely.

Since the major reason an API operation is structured to be asynchronous is that it will take significant time to complete, very frequent polling for completion via calls to **Query Job Status** can lead to significant unproductive use of client application and HMC resources. In order to eliminate the need to poll at all, the Web Services API also provides asynchronous notifications of job completion via its JMS notification capability. IBM recommends that client applications use this notification facility to determine when a job is complete rather than polling for completion. See "Job completion notification" on page 29 for details on this notification mechanism.

If it is not practical for a client application to use asynchronous notification of job completion, the application should introduce elapsed-time delays between successive **Query Job Status** requests to poll for job completion in order to reduce unproductive use of resources.

Query Job Status

The **Query Job Status** operation returns the status associated with an asynchronous job.

HTTP method and URI

GET /api/jobs/{*job-id*}

In this request, the URI variable *{job-id}* is the identifier of an asynchronous job associated with this user, as returned in the response of the operation that initiated the job.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`status`	String Enum	An indication of the current disposition of the job. The possible values are as follows: • **"running"** - indicates that the job was found and has not run to completion at the time of the query. • **"complete"** - indicates that the job was found and has completed running. The successful or error completion of the job is indicated by the **job-status-code** and **job-reason-code** fields.
`job-status-code`	Integer; Field provided only if status is **"complete"**	The job completion status code. This field is provided only if the **status** field is set to **"complete"**. This field provides the major status code describing the success or failure completion of the asynchronous action represented by the job. It is expressed in terms of an HTTP status code (i.e. the HTTP status code that would have been returned for the operation had it been performed synchronously). The values provided here and their meaning depend on the particular action that is being performed asynchronously. The description of these values is provided as part of the description for the operation that initiated the asynchronous job.

Field name	Type	Description
`job-reason-code`	Integer; Field provided only if status is **"complete"**	The job completion reason code. This field is provided only if the **status** field is set to **"complete"** and only if the **job-status-code** field indicates an error completion (status code other than 2XX). When present, this field provides a more detailed reason code describing the success or failure completion of the asynchronous action represented by the job. It is expressed in terms of the API reason code as are returned in standard error response bodies provided by the API. The values provided here and their meaning depend on the particular action that is being performed asynchronously. The description of these values is provided as part of the description or the operation that initiated the asynchronous job.
`job-results`	Object; Field provided only if status is **"complete"**	A nested object that provides results for the job. This field is provided only if the **status** field is set to **"complete"** but is optional even for complete jobs. If the asynchronous operation has no result information (beyond the job status and reason codes) then this field is omitted. The structure of the nested object provided by this field and its meaning depends on the particular action that is being performed asynchronously. The description of this object's structure is provided as part of the description or the operation that initiated the asynchronous job.

Description

The **Query Job Status** operation returns the status associated with an asynchronous job. The results depend on whether the job is still running or is complete.

If the job designated by the URI is still running, the operation sets the **status** field in the response body to **"running"** and provides no other information about the job. The client application may repeat the query at a later time, but should avoid frequent polling since that can lead to unproductive use of client and HMC resources. In order to eliminate the need to poll at all, the client application can (and should) use the asynchronous notifications facility provided by the API to receive notification of job completion via a JMS-based message. See "Job completion notification" on page 29 for details on this notification mechanism.

If the job is complete, the operation sets the **status** field in the response body to **"complete"** and provides the other completion-related fields defined in the Response Body Contents section above to report the results to the client application. Once complete, job status is retained by the HMC for a minimum of 4 hours to allow the client application time to retrieve the results, but this status and results are not held indefinitely.

If the URI does not designate a job associated with the current API session HTTP status code 404 (Not Found) is returned to the client.

Authorization requirements

This operation has the following authorization requirement:

- The user must be correctly authenticated.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 44.

The following HTTP status codes are returned for the indicated operation-specific errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The URI does not designate an asynchronous job associated with the API user.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/jobs/86e44546-107f-11e1-bde0-0010184c8334 HTTP/1.1
x-api-session: 2ltfe2c2q3ti2b2pwq1wfwuzifoi4rymqa8ktzjep7dbyrll0k
```

Figure 5. Query Job Status: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 16 Nov 2011 18:19:35 GMT
content-type: application/json;charset=UTF-8
content-length: 63
{
   "job-reason-code": 0,
   "job-status-code": 200,
   "status": "complete"
}
```

Figure 6. Query Job Status: Response

Delete Completed Job Status

The **Delete Completed Job Status** operation deletes the job status and results associated with a job that has been completed.

HTTP method and URI

DELETE /api/jobs/{*job-id*}

In this request, the URI variable *{job-id}* is the identifier of an asynchronous job associated with this session, as returned in the URI of the operation that initiated the job.

Description

The **Delete Completed Job Status** operation deletes the job status and results associated with a job that has been completed.

If the job designated by the request URI is complete, its completion status and results are deleted from the HMC and status code 204 (No Content) is returned to the client.

If the job is still running, the operation fails and HTTP status code 409 (Conflict) is returned to the client.

If the URI does not designate a job associated with the current API user, or if the job's status has already been deleted (either explicitly, or due to expiration of the status retention interval), HTTP status code 404 (Not Found) is returned to the client.

Authorization requirements

This operation has the following authorization requirement:
- The user must be correctly authenticated.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated operation-specific errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The URI does not designate an asynchronous job associate with this session.
409 (Conflict)	40	The URI designates an asynchronous job that is not yet complete.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Usage notes

- This operation is defined to operate only on jobs that have been completed, i.e. have a **status** field with value **"complete"**. As a result, this operation cannot be used to cancel an in-progress asynchronous operation. The ability to cancel an in-progress asynchronous operation is not provided by the API.
- Once an asynchronous job is complete, job status is retained by the HMC for a minimum of 4 hours to allow the client application time to retrieve the results, but this status and results are not held indefinitely. At the expiration of the retention interval, job status is deleted as if the Delete Complete Job Status operation were called.

Example HTTP interaction

```
DELETE /api/jobs/86e44546-107f-11e1-bde0-0010184c8334 HTTP/1.1
x-api-session: 2ltfe2c2q3ti2b2pwq1wfwuzifoi4rymqa8ktzjep7dbyrll0k
```

Figure 7. Delete Completed Job Status: Request

```
204 No Content
date: Wed, 16 Nov 2011 18:19:35 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 8. Delete Completed Job Status: Response

Chapter 7. Ensemble composition

A zEnterprise ensemble is a grouping of one or more computing systems, referred to as nodes, that are managed in a coordinated way for purposes of virtualization and workload management. Currently, the ensemble can be comprised of zEnterprise (or later) Central Processing Complexes (CPCs) and their associated system resources. Each such node consists of the traditional System z elements (processors, memory, I/O, LPARs) along with an optional processor feature termed a zEnterprise Blade Extension (zBX) that provides blade-based computing resources for the node. In addition to the per-member resources, the ensemble also encompasses certain ensemble-wide resources that are shared by all of the members of the ensemble, including a secure, platform-managed data network.

An ensemble will typically consist of at least one node. However, when initially created, an ensemble starts out with no nodes, so an empty ensemble is a legitimate configuration that may sometimes exist. CPCs are explicitly added as nodes to, or removed from, an ensemble using operations provided in the API or using management tasks on the HMC UI.

Not all models of System z processors support ensembles and ensemble-based management. Platform support for ensembles and ensemble-based management was first delivered with the zEnterprise system family, and thus initially only zEnterprise models can participate as ensemble nodes. Follow-on processor models to zEnterprise may also support ensemble management.

For purposes of controlling the configuration and operational state of the elements of an ensemble, each ensemble is managed from a customer-designated primary/alternate pair of Hardware Management Consoles (HMC) that is referred to as the ensemble HMCs for that ensemble. At any one point in time, only one HMC of the pair is in an active primary role, with the second of the pair acting strictly as a passive backup or alternate. The primary ensemble HMC may manage at most one ensemble. The primary ensemble HMC is the component that provides the main administrative user interface for the ensemble, and it is the component that acts as the access point for the majority of the Web Services API. However, a selected set of API function is also available from the alternate HMC.

For purposes of this definition, it is usually sufficient to focus on the primary ensemble HMC only, except when specifically considering aspects that pertain to the setup of the primary/alternate pair or handling the failover mechanism. Therefore, use of the term “ensemble HMC” without further primary or alternate qualification should be interpreted as indicating the primary ensemble HMC.

Note: Properties related to the identity and addressing of the alternate HMC for the ensemble managed by the current HMC are provided as part of the data model for the Console object.

Ensemble composition operations summary

The following tables provide an overview of the ensemble composition operations provided.

Table 8. Ensemble composition: operations summary

Operation name	HTTP method and URI path
“List Ensembles” on page 53	`GET /api/ensembles`
“Get Ensemble Properties” on page 55	`GET /api/ensembles/{ensemble-id}`
“Update Ensemble Properties” on page 57	`POST /api/ensembles/{ensemble-id}`

Table 8. Ensemble composition: operations summary (continued)

Operation name	HTTP method and URI path
"List Ensemble Nodes" on page 59	`GET /api/ensembles/{ensemble-id}/nodes`
"Get Node Properties" on page 61	`GET /api/ensembles/{ensemble-id}/nodes/{node-id}`
"Add Node (CPC) to Ensemble" on page 63	`POST /api/ensembles/{ensemble-id}/nodes`
"Remove Node from Ensemble" on page 65	`DELETE /api/ensembles/{ensemble-id}/nodes/{node-id}`

Table 9. Ensemble composition: URI variables

Variable	Description
{ensemble-id}	Object ID (UUID) of an Ensemble object
{node-id}	Element ID of a node of an Ensemble.

Ensemble object

An ensemble object represents a single zEnterprise ensemble.

Data Model

This object includes the properties defined in the "Base managed object properties schema" on page 33, including the operational-status properties, with the following class-specific specialization:

Table 10. Ensemble object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path for an ensemble object is of the form `/api/ensembles/{ensemble-id}` where *{ensemble-id}* is the value of the **object-id** property of the ensemble object.
parent	—	String/ URI	An ensemble object is conceptually a root object and has no parent, so this property is always null.
class	—	String	The class of an ensemble object is **"ensemble"**.
name	(w)(pc)	String (1-16)	The display name specified for the ensemble. Alphanumeric, space, or any of "+<=>%&*\'"(),_-./:;?" are valid characters.
description	(w)(pc)	String (0-128)	Text describing the Ensemble. Alphanumeric, space, or any of "+<=>%&*\'"(),_-./:;?" are valid characters.
status	(sc)	String Enum	The status of the ensemble representing the current communication status between the primary and alternate HMC: • **"alternate-communicating"** – The primary is communicating to the alternate • **"alternate-not-communicating"** - The primary is not communicating to the alternate
additional-status	—	String Enum	An ensemble object has no additional-status.

Class specific additional properties

In addition to the properties defined in included schemas, this object includes the following additional class-specific properties:

Table 11. Ensemble composition: class specific properties

Name	Qualifier	Type	Description
management-enablement-level	(pc)	String Enum	The zManager management enablement level for this ensemble. The level determines which zManager advanced management functions are available for use. Values: • **"manage"**- Gives you the basics for managing an ensemble. It includes HMC operational controls for IBM zEnterprise BladeCenter Extension (zBX), change management of firmware across the ensemble, energy monitoring, virtual networking with automated provisioning, virtual server management, and a base level of performance management. • **"automate"**- Provides more leverage from the ensemble by managing workloads and energy. This level of support includes power capping, power savings mode, and platform performance management.
cpu-perf-mgmt-enabled-power-vm	(w)(pc)	Boolean	If true, management of processor performance is enabled for PowerVM® virtualization hosts. Management of processor performance is also available for virtual servers. Performance management properties may be updated if the ensemble management-enablement-level is **"automate"**.
cpu-perf-mgmt-enabled-zvm	(w)(pc)	Boolean	If true, management of processor performance is enabled for z/VM virtualization hosts. Management of processor performance is also available for virtual servers. Performance management properties may be updated if the ensemble management-enablement-level is **"automate"**.
unique-local-unified-prefix	(pc)	String	Unique local address (ULA) prefix applied to addresses used for management communication between virtual servers and their virtualization hosts. The ULA prefix of the form `fdXX:XXXX:XXXX::/48` is formed by substituting the `X`'s with a pseudo-random 40-bit global ID using the algorithm defined in RFC 4193. The prefix may not be updated through the API.
load-balancing-enabled	(w)(pc)	Boolean	If true, Load Balancing is enabled for this ensemble. Load Balancing properties may be updated if the ensemble management-enablement-level is **"automate"**.
load-balancing-port	(w)(pc)	Integer (1024-65535)	The Load Balancing port value in the range 1024-65534. The default port is 3860. Load Balancing properties may be updated if the ensemble management-enablement-level is **"automate"**.
load-balancing-ip-addresses	(w)(pc)	Array of String	The IPV4 address or IPV6 addresses of Load Balancers allowed access to the Load Balancing function. The strings are in dotted-decimal form (“nnn.nnn.nnn.nnn”) for IPV4 addresses. The strings are in eight groups of four hexadecimal digits separated by colons (e.g. `2001:0db8:85a3:0000:0000:8a2e:0370:7334`), for IPV6 addresses. Load Balancing properties may be updated if the ensemble management-enablement-level is **"automate"**.

Table 11. Ensemble composition: class specific properties (continued)

Name	Qualifier	Type	Description
mac-prefix	(pc)	String (2)	The Prefix Address *xx*:00:00:00:00:00/8 is the ensemble mac prefix. All Mac addresses dynamically generated by zManager will be within the ensemble mac prefix. *xx* defaults to 02. The MAC prefix may not be updated through the API.
reserved-mac-address-prefixes	(pc)	Array of objects	The list of reserved MAC address prefixes. Each reserved MAC address prefix will be an object in the form of a MAC address prefix nested object, as described in "MAC address prefix nested object." MAC address prefixes may not be updated through the API.

Energy management related additional properties: In addition to the properties defined above, this object includes the following additional class-specific properties related to energy management. For further explanation of the various states involved, please see "Special states" on page 138,

Table 12. Ensemble composition: energy management related additional properties

Name	Qualifier	Type	Description
power-rating	—	Integer	Specifies the maximum power usage in watts (W) of this ensemble. This is a calculated value as indicated by the electrical rating labels or system rating plates of the ensemble components.
power-consumption	(mg)	Integer	Specifies the current power consumption in watts (W) for this ensemble.

MAC address prefix nested object: A MAC address prefix object is a nested object of an ensemble object. An ensemble may contain zero or more MAC address prefixes. MAC address prefix properties may not be updated through the API.

The following properties are supported:

Table 13. Ensemble composition: MAC address prefix nested object related additional properties

Field name	Type	Description
mac-address	String	The MAC address represented as 6 groups of two hexadecimal digits separated by colons, e.g. `01:23:45:67:89:ab`. The MAC address uses the ensemble prefix.
prefix-length	Integer	The bit length of the MAC address prefix. This is a 2-digit value with these parameters in the range 12-44.

Node object

A node is an element of the ensemble object that represents a system that is currently a member of the ensemble. Each node contains the following properties:

Table 14. Ensemble composition: node properties

Field name	Type	Description
element-uri	String/ URI	The URI path for a node of an ensemble is of the form `/api/ensembles/{`*`ensemble-id`*`}/nodes/{`*`node-id`*`}` where *{ensemble-id}* is the value of the **object-id** property of the ensemble, and *{node-id}* is a locally unique element ID for the node. For nodes of **type "cpc"**, *{node-id}* is the value of the **object-id** property of the CPC that is represented by the node.
parent	String/ URI	The canonical URI path of the ensemble containing this node.

Table 14. Ensemble composition: node properties (continued)

Field name	Type	Description
class	String	The value **"node"**.
type	String Enum	The type of node. Currently this is always the value **"cpc"**.
member	String/ URI	The canonical URI path of the system element that is represented as a member of the ensemble by this object. For nodes of type **"cpc"**, this is the URI path of the CPC object.

Operations

List Ensembles

The **List Ensembles** operation lists the ensembles managed by the HMC.

HTTP method and URI

GET /api/ensembles

Query parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects. If matches are found, the response will be an array with all ensembles that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
ensembles	Array of objects	Array of nested ensemble-info objects (described in the next table). If the HMC is not a primary HMC, an empty array is provided.

Each nested ensemble-info object contains the following fields:

Field name	Type	Description
object-uri	String/ URI	Canonical URI path of the Ensemble object, in the form /api/ensembles/{*ensemble-id*}.
name	String	Display name of the Ensemble object.
status	String Enum	The **status** property of the Ensemble object.

Description

This operation lists the ensembles that are managed by this HMC. The object URI, display name, and status are provided for each.

If the **name** query parameter is specified, the returned list is limited to those ensembles that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

An ensemble is included in the list only if the API user has object-access permission for that object. If an HMC is a manager of an ensemble but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the HMC does not manage any ensembles, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to any ensemble object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 53.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

This operation, as well as other aspects of the Web Services API, has been structured to allow for the possibility that an HMC might be the manager of multiple ensembles. However, in the current implementation, an HMC can be the manager of at most one ensemble, so the resulting list of ensembles will contain either zero or one entries.

Example HTTP interaction

```
GET /api/ensembles HTTP/1.1
x-api-session: 1f2g70m2e9b4sawt53ydp1oc9nzzs4sduc2wr2bzgmy5dhs7pz
```

Figure 9. List Ensembles: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 20 Jul 2011 18:41:03 GMT
content-type: application/json;charset=UTF-8
content-length: 207
{
   "ensembles": [
      {
         "name": "R32Ensemble",
         "object-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
         "status": "alternate-communicating"
      }
   ]
}
```

Figure 10. List Ensembles: Response

Get Ensemble Properties

The **Get Ensemble Properties** operation retrieves the properties of a single Ensemble object that is designated by its object ID.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object for which properties are to be obtained.

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the ensemble object as defined in the Data Model section above. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The operation returns the current properties for the ensemble object specified by *{ensemble-id}*.

On successful execution, all of the current properties as defined by the Data Model for the ensemble object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the Ensemble object designated by *{ensemble-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is provided and the response body is as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object-id in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object-access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026 HTTP/1.1
x-api-session: 1f2g70m2e9b4sawt53ydp1oc9nzzs4sduc2wr2bzgmy5dhs7pz
```

Figure 11. Get Ensemble Properties: Request

```
200 OK
x-request-id: Sx3a Rx1
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 20 Jul 2011 18:41:03 GMT
content-type: application/json;charset=UTF-8
content-length: 959
{
   "acceptable-status": [
      "alternate-communicating"
   ],
   "class": "ensemble",
   "cpu-perf-mgmt-enabled-power-vm": true,
   "cpu-perf-mgmt-enabled-zvm": true,
   "description": "",
   "has-unacceptable-status": false,
   "is-locked": false,
   "load-balancing-enabled": true,
   "load-balancing-ip-addresses": [
      "1.1.1.1"
   ],
   "load-balancing-port": 9876,
   "mac-prefix": "02:00:00:00:00:00",
   "management-enablement-level": "automate",
   "name": "R32Ensemble",
   "object-id": "87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "object-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "parent": null,
   "power-consumption": 8311,
   "power-rating": 46522,
   "reserved-mac-address-prefixes": [
      {
         "mac-address": "02:c1:00:00:00:00",
         "prefix-lenght": 16
      }
   ],
   "status": "alternate-communicating",
   "unique-local-unified-prefix": "fd07:8c9:1ba3:0:0:0:0:0"
}
```

Figure 12. Get Ensemble Properties: Response

Update Ensemble Properties

The **Update Ensemble Properties** operation modifies simple writeable properties of an ensemble object.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object for which properties are to be updated.

Request body contents

The request body is expected to contain a JSON object with one or more of the following fields. Only fields that are being changed are necessary to supply.

Field name	Type	Description
name	String (1-16)	The new name to give the ensemble, as described in "Data Model" on page 50.
description	String (0-128)	The new description to give the ensemble, as described in "Data Model" on page 50.
cpu-perf-mgmt-enabled-power-vm	Boolean	The PowerVM virtualization host processor performance management enablement setting, as described in "Data Model" on page 50.
cpu-perf-mgmt-enabled-zvm	Boolean	The z/VM virtualization host processor performance management enablement setting, as described in "Data Model" on page 50.
load-balancing-enabled	Boolean	The Load Balancing enablement setting for this ensemble, as described in "Data Model" on page 50.
load-balancing-port	Integer	The Load Balancing port for this ensemble, as described in "Data Model" on page 50.
load-balancing-ip-address	Array of Strings	The Load Balancing ip addresses, as described in "Data Model" on page 50.

Description

This operation updates one or more simple writeable properties of the ensemble object identified by *{ensemble-id}*.

On successful execution, the ensemble object has been updated with the supplied property values and status code 204 (No Content) is returned without supplying a response body.

If the update changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

The URI path must designate an ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have permission to the Ensemble Details task as well, otherwise status code 403 (Forbidden) is returned.

The request body is validated against the schema described in the Request Body Contents section. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the Ensemble object designated by *{ensemble-id}*
- Action/task permission to the **Ensemble Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
	228	The ensemble's **management-enablement-level** property does not allow the updating of a property specified in the request body.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	3	The operation cannot be performed because the object designated by the request URI is currently locked.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334 HTTP/1.1
x-api-session: 297n8iun1251svgcju9tvsai0rrew4ieawx97ykucbxy69bwr2
content-type: application/json
content-length: 34
{
   "name": "SS-Ensemble-1"
}
```

Figure 13. Update Ensemble Properties: Request

```
204 No Content
date: Wed, 07 Dec 2011 04:54:01 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 14. Update Ensemble Properties: Response

List Ensemble Nodes

The **List Ensemble Nodes** operation lists the nodes of an ensemble managed by the HMC.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/nodes

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object for which nodes are to be listed.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`nodes`	Array of objects	Array of nested node-info objects (described in the next table). If the ensemble has no nodes, an empty array is provided.

Each nested node-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	URI path of the ensemble node object, in the form `/api/ensembles/{ensemble-id}/nodes/{node-id}`. For a node of **type "cpc"**, the node object represents a CPC as a member of the node, and the *{node-id}* component of this URI path is the **object-id** property of the underlying CPC object.
`type`	String Enum	The **type** of the ensemble node. Currently this is always **"cpc"**.
`name`	String	The **name** property of the underlying object that is represented by the node
`status`	String Enum	The **status** property of the underlying object that is represented by the node

Description

This operation lists the nodes of an ensemble specified by its *{ensemble-id}*. The element URI and type are provided for each.

A node is included in the list only if the API user has object-access permission to the underlying object that is represented by the node. For a node of **type "cpc"**, the underlying object is a CPC. If an ensemble contains a node but the API user does not have permission to the related object, that object is simply omitted from the list but no error status code results.

If the ensemble is empty (has no nodes), an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the ensemble designated in the request URI
- Object access permission to the underlying object that is represented by a node included in the result. For nodes of **type "cpc"**, the underlying object is a CPC.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes HTTP/1.1
x-api-session: 1f2g70m2e9b4sawt53ydp1oc9nzzs4sduc2wr2bzgmy5dhs7pz
```

Figure 15. List Ensemble Nodes: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 20 Jul 2011 18:41:03 GMT
content-type: application/json;charset=UTF-8
content-length: 250
{
   "nodes": [
      {
         "element-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-
       8466- be79e49dd340",
         "name": "R32",
         "status": "operating",
         "type": "cpc"
      }
   ]
}
```

Figure 16. List Ensemble Nodes: Response

Get Node Properties

The **Get Node Properties** operation retrieves the properties of a single Node object.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/nodes/{*node-id*}

URI variables

Variable	Description
{ensemble-id}	Object ID of the ensemble containing the node for which properties are to be obtained.
{node-id}	Element ID of the node which for which properties are to be obtained.

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the node object as defined in "Data Model" on page 50. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The operation returns the current properties for the node object specified by the request URI.

On successful execution, all of the current properties as defined by the Data Model for the ensemble object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing node object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the ensemble designated in the request URI
- Object access permission to the underlying object that is represented by a node included in the result. For nodes of **type "cpc"**, the underlying object is a CPC.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is provided and the response body is as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.
	225	The element-id in the URI (*{node-id}*) does not designate an existing node of the ensemble designed by *{ensemble-id}*.
	227	The element-id in the URI (*{node-id}*) does not designate an existing object, or the API user does not have access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-e79e49dd340 HTTP/1.1
x-api-session: 1f2g70m2e9b4sawt53ydp1oc9nzzs4sduc2wr2bzgmy5dhs7pz
```

Figure 17. Get Node Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 20 Jul 2011 18:41:03 GMT
content-type: application/json;charset=UTF-8
content-length: 295
{
   "class": "node",
   "element-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
    be79e49dd340",
   "member": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "type": "cpc"
}
```

Figure 18. Get Node Properties: Response

Add Node (CPC) to Ensemble

The **Add Node to Ensemble** operation adds a CPC to an ensemble, creating a new Node object to represent the membership as a result.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/nodes

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object to which the CPC is to be added as a node.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
cpc	String/URI	The canonical URI path identifying the CPC to be added to the targeted ensemble as a new node.

Response body contents

Field name	Type	Description
node-uri	String/URI	The canonical URI path identifying the node that was created in the targeted ensemble, in the form /api/ensembles/{*ensemble-id*}/nodes/{*node-id*}.

Description

This operation adds a CPC to the ensemble targeted by the request URI and creates a new Node object to represent the membership. Refer to the *zEnterprise System Ensemble Planning and Configuring Guide* for details on managing members of an ensemble.

Upon successful completion, HTTP status code 201 (Created) is returned and the response body includes the URI of the node that was created to represent the membership. This URI is also provided as the value of the **Location** header in the response.

If the CPC is already a node of an ensemble (either the targeted ensemble or another one) HTTP status code 400 (Bad Request) is returned with associated reason code 224.

The URI path must designate an existing ensemble and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action/task permission to the Add Member to Ensemble task as well, otherwise status code 403 (Forbidden) is returned. Additionally if the CPC is ineligible to be added to the ensemble a status code 409 (Conflict) is returned.

The request body is validated against the schema described in "Request body contents" on page 63. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the ensemble object passed in the request URI
- Object access permission to the CPC object passed in the request body.
- Action/task permission to the Add Member to Ensemble task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents" on page 63.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	221	The operation cannot be performed because the CPC to be added as a member is not an eligible machine model.
	222	The operation cannot be performed because the CPC does not have the Ensemble without zBX Feature or the zBX Feature installed.
	224	The operation cannot be performed because the CPC is a member of another ensemble or already a member of the ensemble targeted in the request URI.
	229	The operation cannot be performed because the maximum number of CPCs would be exceeded for the ensemble.
	230	A CPC may be added to an Ensemble only, if the LICCC QoS value of the CPC is the same as the aggregated QoS of the Ensemble.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.
	227	The object-id in the URI of the CPC object in the request body does not designate an existing object, or the API user does not have access permission to it.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	3	The operation cannot be performed because the object designated by the request URI is currently locked to prevent disruptive changes from being made.
	220	The operation cannot be performed because the CPC to be added as a member is currently busy performing some other operation.
	223	The operation cannot be performed because the CPC to be added as a member is not in the correct state to be added to the ensemble. The CPC cannot be added if the CPC operational status is **"not-communicating"**.
	226	The operation cannot be performed because the CPC object to be added as a member is currently locked to prevent disruptive changes from being made.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Remove Node from Ensemble

The **Remove Node from Ensemble** operation removes a node from an ensemble.

HTTP method and URI

```
DELETE /api/ensembles/{ensemble-id}/nodes/{node-id}
```

URI variables

Variable	Description
{ensemble-id}	Object ID of the ensemble from which the targeted node is to be removed as a member.
{node-id}	Element ID of the node which is to be removed from the targeted ensemble.

Description

This operation removes a specified node from the specified ensemble. The node is identified by the *{node-id}* variable in the URI, and the ensemble is identified by the *{ensemble-id}* variable in the request URI.

Refer to the *zEnterprise System Ensemble Planning and Configuring Guide* for details on managing members of an ensemble.

Upon successfully removing the node as a member, HTTP status code 204 (No Content) is returned.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing node element. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have Remove Member from Ensemble action/task permission as well, otherwise status code 403 (Forbidden) is returned. Additionally if the CPC is ineligible to be removed from the ensemble a status code 409 (Conflict) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the ensemble object passed in the request URI

- Object access permission to the system element represented by the node element. For nodes of **type "cpc"**, this is a CPC object
- Action/task permission to the **Add Member to Ensemble** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	225	The operation cannot be performed because the node (*{node-id}*) to be removed is not a member of the ensemble (*{ensemble-id}*) designated by the request URI.
	231	The operation cannot be performed because the node (*{node-id}*) to be removed has entitled blades
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.
	227	The element-id in the URI (*{node-id}*) does not designate an existing object, or the API user does not have access permission to it.
409 (Conflict)	2	The operation cannot be performed because the object (*{ensemble-id}*) designated by the request URI is currently busy performing some other operation.
	3	The operation cannot be performed because the object (*{ensemble-id}*) designated by the request URI is currently locked to prevent disruptive changes from being made.
	220	The operation cannot be performed because the node (*{node-id}*) to be removed as a member is currently busy performing some other operation.
	223	The operation cannot be performed because the node (*{node-id}*) to be removed as a member is not in the correct state to be removed to the ensemble. For nodes of **type "cpc"**, the node cannot be removed if the CPC operational status is **"not-communicating"**.
	226	The operation cannot be performed because the node to be removed as a member is currently locked to prevent disruptive changes from being made.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Inventory service data

Information about the Ensembles managed by the HMC and the associated nodes can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for ensemble and node objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by class, implicitly via a containing

category, or by default) that objects of class "ensemble" are to be included. Information for a particular ensemble (and associated node) is included only if the API user has object-access permission to that object.

For each ensemble to be included, the inventory response array includes the following:

- An array entry for the ensemble object itself. This entry is a JSON object with the same contents as is specified in the Response Body Contents section for the **Get Ensemble Properties** operation. That is, the data provided is the same as would be provided if a **Get Ensemble Properties** operation were requested targeting this object.
- An array entry for each node associated with the ensemble. For each such node, an entry is included that is a JSON object with the same contents as specified in the Response Body Contents section of the **Get Ensemble Properties** operation. As a result, the data provided is the same as would be obtained if a **Get Node Properties** operation were requested for each node listed by a **List Ensemble Nodes** operation targeting the ensemble.

The array entry for an ensemble object will appear in the results array before entries for associated nodes.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the Get Inventory response to describe a ensemble (named "R32Ensemble") with a single node as member. These objects would appear as a sequence of array entries in the response array:

```
{
   "acceptable-status": [
      "alternate-communicating"
   ],
   "class": "ensemble",
   "cpu-perf-mgmt-enabled-power-vm": false,
   "cpu-perf-mgmt-enabled-zvm": true,
   "description": "FVT Test",
   "has-unacceptable-status": false,
   "is-locked": false,
   "load-balancing-enabled": true,
   "load-balancing-ip-addresses": [
      "1.1.1.1"
   ],
   "load-balancing-port": 9876,
   "mac-prefix": "02:00:00:00:00:00",
   "management-enablement-level": "automate",
   "name": "R32Ensemble",
   "object-id": "87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "object-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "parent": null,
   "power-consumption": 8298,
   "power-rating": 46288,
   "reserved-mac-address-prefixes": [],
   "status": "alternate-communicating",
   "unique-local-unified-prefix": "fd07:8c9:1ba3:0:0:0:0:0"
},
{
   "class": "node",
   "element-uri": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
   be79e49dd340",
   "member": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "type": "cpc"
}
```


Figure 19. Ensemble object: Sample inventory data

Usage notes

When configured as recommended by IBM, the process of recovering from the failure of the primary ensemble-management HMC by takeover by the alternate HMC includes movement of the IP address of the former primary to the new primary. When this occurs, explicit redirection of API requests to the newly designated primary HMC is not needed. However, the IP address swapping may not be possible in certain network configurations. The address of the alternate ensemble-management HMC for the current Ensemble is provided as properties of the Console object (representing the current HMC) to allow applications to explicitly redirect requests to the other HMC of the pair in these cases.

Chapter 8. zBX infrastructure elements

A zEnterprise ensemble is a grouping of one or more CPCs and their associated system resources that are managed in a coordinated way for purposes of virtualization and workload management. Each member of the ensemble may have a zEnterprise Blade Extension (zBX) attached to the CPC to provide blade-based computing resources for the member.

The zBX has the following key components: BladeCenter chassis, Blades and VLAN capable switches all mounted in dedicated racks. The zBX is connected to the CPC through two dedicated integrated networks. Within the ensemble, advanced management capabilities are provided for the blades housed within the zBX. The zBX components are configured, managed and serviced in the same way as the other components of the CPC. Management of the zBX is provided only if the associated CPC is a member of an Ensemble.

zBX physical network overview

The zBX contains physical network switches that provide the connectivity between the blades and CPCs in the zEnterprise Intra-Ensemble Data Network (IEDN). There are two types of Ethernet switches within each zBX:

- Top-of-Rack Switches (TORs) – A pair of TORs reside in each zBX and act as a primary and backup. TORs connect the blades in the zBX to System z network interfaces, and to other external networking equipment, such as routers.
- Ethernet Switch Modules (ESMs) – A pair of ESMs reside in each BladeCenter chassis and connect the blades in the zBX to the IEDN and provide the links to the TORs.

The initial configuration and setup of the physical switches are provided by zManager. The ESMs are not accessible for configuration changes through the Web Services API, however, performance metrics are provided through the Metrics Service. Some TOR management will typically be required by an external administrator; therefore, when managing virtual networks, administrators must consider requirements for configuring the Top-of-Rack switch ports. Only certain types of TOR ports support configuration by the zManager user, these are:

- External - Ports that connect to a customer's external network.
- Internal - Ports are internal ports that extend to the ESMs which connect to the blades. The configuration of this type of port is intended only to support the ISAOPT Coordinator. In this case, VLAN- tagging is required to allow traffic from ISAOPT to System z to be tagged with the proper VLAN ID.

The configuration properties supported by these ports are:

- Virtual networks - The following port types can be configured to a virtual network:
  - External
  - Internal
- MAC Filters – The following port types support MAC filters:
  - External

zBX infrastructure operations summary

The following tables provide an overview of the operations provided.

Table 15. zBX infrastructure: operations summary

Operation name	HTTP method and URI path
"List zBXs of a CPC" on page 72	`GET /api/cpcs/{cpc-id}/zbxs`
"List zBXs of a Ensemble" on page 74	`GET /api/ensembles/{ensemble-id}/zbxs`
"Get zBX Properties" on page 76	`GET /api/zbxs/{zbx-id}`
"List Top-of-Rack Switches of a zBX" on page 81	`GET /api/zbxs/{zbx-id}/top-of-rack-switches`
"Get Top-of-Rack Switch Properties" on page 83	`GET /api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}`
"Get Top-of-Rack Switch Port Details" on page 85	`GET /api/zbxs/{zbx_id}/top-of-rack-switches/{tor-id}/ports/{port-id}`
"Update Top-of-Rack Switch Port Properties" on page 87	`POST/api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}`
"Add MAC Filters to Top-of-Rack Switch Port" on page 89	`POST /api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}/operations/add-mac-filters`
"Remove MAC Filters from Top-of-Rack Switch Port" on page 91	`POST /api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}/operations/remove-mac-filters`
"Add Top-of-Rack Switch Port to Virtual Networks" on page 93	`POST /api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}/operations/add-virtual-networks`
"Remove Top-of-Rack Switch Port from the Virtual Networks" on page 95	`POST /api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}/operations/remove-virtual-networks`
"List Racks of a zBX" on page 98	`GET /api/zbxs/{zbx-id}/racks`
"Get Rack Properties" on page 100	`GET /api/racks/{rack-id}`
"List BladeCenters in a Rack" on page 105	`GET /api/racks/{rack-id}/bladecenters`
"List BladeCenters in a zBX" on page 107	`GET /api/zbxs/{zbx-id}/bladecenters`
"Get BladeCenter Properties" on page 109	`GET /api/bladecenters/{bladecenter-id}`
"List Blades in a BladeCenter" on page 117	`GET /api/bladecenters/{bladecenter-id}/blades`
"List Blades in a zBX" on page 119	`GET /api/zbxs/{zbx-id}/blades`
"Get Blade Properties" on page 122	`GET /api/zbxs/{zbx-id}/blades`
"Activate a Blade" on page 126	`POST /api/blades/{blade-id}/operations/activate`

Table 15. zBX infrastructure: operations summary (continued)

Operation name	HTTP method and URI path
"Deactivate a Blade" on page 128	`POST /api/blades/{blade-id}/operations/deactivate`
"Create IEDN Interface for a DataPower XI50z Blade" on page 130	`POST /api/blades/{blade-id}/iedn-interface`
"Delete IEDN Interface for a DataPower XI50z Blade" on page 133	`DELETE /api/blades/{blade-id}/iedn-interface/{iedn-interface-id}`

Table 16. zBX infrastructure: URI variables

Variable	Description
{cpc-id}	Object ID of a CPC
{zbx-id}	Object ID of a zBX
{tor-id}	Element ID of a TOR object
{port-id}	Element ID of a TOR port
{rack-id}	Object ID of a Rack
{bladecenter-id}	Object ID of a BladeCenter
{blade-id}	Object ID of a Blade
{iedn-interface-id}	Element ID of an IEDN Interface
{hardware-message-id}	Element ID of a Hardware Message

zBX object

A zBX object represents the single zBX of a CPC. There is at most one zBX object for each CPC that is a member of an ensemble.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, but does not provide the operational-status-related properties defined in that schema because it does not maintain the concept of an operational status.

The following class-specific specializations apply to the other Base Managed Object properties:

Table 17. zBX object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
name	(ro)	String (1-64)	The name of the ZBX. Currently assigned by zManager as 21 characters of the form <Machine type>-<machine model>-<machine serial number> of the zBX object.[1]
description	—	String	This field is not provided.
object-uri	—	String/ URI	The canonical URI path for a zBX object is of the form `/api/zbxs/{zbx-id}`.
parent	—	String/ URI	The canonical URI path of the parent CPC object.
class	—	String	The value **"zbx"**.

Table 17. zBX object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
Note: [1] This **name** property is currently assigned by zManager and is not writeable. However, it is possible that the API could be extended to allow this property to be writeable, in which case an API or User-Interface user could change the name to contain arbitrary data. Therefore, API client applications should not rely on the contents and format of the **name** property always being in the form of the zManager-assigned name.			

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 18. zBX object: class specific properties

Name	Qualifier	Type	Description
machine-type	—	String	4 characters.
machine-model	—	String	3 characters.
machine-serial	—	String	12 characters.
current-isaopt-entitlements	—	Integer	Number of blades entitled as ISAOPT blades.
max-isaopt-entitlements	—	Integer	Maximum licensed to be entitled as ISAOPT blades.
current-power-entitlements	—	Integer	Number of blades entitled as POWER® IBM blades.
max-power-entitlements	—	Integer	Maximum licensed to be entitled as POWER IBM blades.
current-systemx-entitlements	—	Integer	Number of blades entitled as System x® IBM blades.
max-systemx-entitlements	—	Integer	Maximum licensed to be entitled as System x IBM blades.
current-dpxi50z-entitlements	—	Integer	Number of blades entitled as DPXI50Z blades.
max-dpxi50z-entitlements	—	Integer	Maximum licensed to be entitled as DPXI50Z blades.

Operations

List zBXs of a CPC

The **List zBXs of a CPC** operation lists the zBXs associated with the CPC. The CPC must be a member of an ensemble.

HTTP method and URI

```
GET /api/cpcs/{cpc-id}/zbxs
```

In this request, the URI variable *{cpc-id}* is the object ID of the CPC object whose zBXs are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
zbxs	Array of objects	Array of nested zbx-info objects, described in the next table. If the CPC does not have a zBX associated with it, an empty array is provided.

Each nested zbx-info object contains the following fields:

Field name	Type	Description
object-uri	String/ URI	Canonical URI path of the zBX object in the form /api/zbxs/*{zbx-id}*.
name	String	The **name** property of the zBX object (based on its machine type, machine model, and machine serial number).

Description

The **List zBXs of a CPC** operation lists the zBXs that are associated with this CPC. The object URI and name are provided for each zBX.

If the **name** query parameter is specified, then a zBX is included in the list only if the name pattern matches the **name** property of the object.

A zBX is included in the list only if the API user has object-access permission for the CPC with which the zBX is associated. If the HMC is a manager of a zBX, but the API user does not have permission to it, that object is omitted from the list, but no error status code results.

If the CPC does not have a zBX, an empty list is provided and the operation completes successfully. Note that if the CPC is not a member of an ensemble, it cannot have a zBX. Therefore, an empty list is provided.

Authorization requirements

This operation has the following authorization requirement:
- Object-access permission to the CPC object specified by the request URI.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	241	CPC object (*{cpc-id}*) is not a member of an ensemble.
404 (Not Found)	1	The object ID *{cpc-id}* does not designate an existing CPC object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- This operation has been structured to allow for the possibility that an HMC might be the manager of multiple zBXs on a CPC. However, in the current implementation, each CPC can have at most one zBX. Therefore, the resulting list of zBXs will contain either zero or one entry.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/zbxs HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 20. List zBXs of a CPC: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:48:58 GMT
content-type: application/json;charset=UTF-8
content-length: 160
{
   "zbxs": [
      {
         "name": "2458-002-0000000ZBX26",
         "object-uri": "/api/zbxs/54a9716c-a326-11e0-9469-001f163805d8"
      }
   ]
}
```

Figure 21. List zBXs of a CPC: Response

List zBXs of a Ensemble

The **List zBXs of a Ensemble** operation lists the zBXs, which are associated with each CPC in the ensemble.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/zbxs

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object whose zBXs are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
zbxs	Array of objects	Array of nested zbx-info objects, described in the next table. If the ensemble does not have any CPCs or zBXs associated with it, an empty array is provided.

Each nested zbx-info object contains the following fields:

Field name	Type	Description
object-uri	String/ URI	Canonical URI path of the zBX object in the form /api/zbxs/*{zbx-id}*.
name	String	The **name** property of the zBX object (based on its machine type, machine model, and machine serial number).

Description

The **List zBXs of a Ensemble** operation lists the zBXs that are part of the ensemble. The object URI and name are provided for each zBX.

If the **name** query parameter is specified, then a zBX is included in the list only if the name pattern matches the **name** property of the object.

A zBX is included in the list only if the API user has object-access permission for the CPC object with which it is associated. If the HMC is a manager of a zBX, but the API user does not have permission to the CPC with which the zBX is associated, that object is omitted from the list, but no error status code results.

If the HMC does not manage any zBXs, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the CPC object passed in the request URI
- Object-access permission to the CPC objects with which a zBX is associated.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID *{ensemble-id}* does not designate an existing ensemble object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/zbxs HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 22. List zBXs of a Ensemble: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:48:58 GMT
content-type: application/json;charset=UTF-8
content-length: 160
{
   "zbxs": [
      {
         "name": "2458-002-0000000ZBX26",
         "object-uri": "/api/zbxs/54a9716c-a326-11e0-9469-001f163805d8"
      }
   ]
}
```

Figure 23. List zBXs of a Ensemble: Response

Get zBX Properties

The **Get zBX Properties** operation retrieves the properties of a single zBX object that is designated by its object ID.

HTTP method and URI

GET /api/zbxs/{*zbx-id*}

In this request, the URI variable *{zbx-id}* is the object ID of the zBX object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the zBX object as defined in the "Data model" on page 71. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get zBX Properties** operation returns the current properties for the zBX object specified by *{zbx-id}*.

On successful execution, all of the current properties as defined in "Data model" on page 71 for the storage resource object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing zBX object and the API user must have object-access permission to the CPC with which the zBX is associated.

Authorization requirements

This operation has the following authorization requirement:
- Object-access permission to the CPC objects with which the zBX is associated.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 76.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{zbx-id}* does not designate an existing zBX object, or the API user does not have object access permission to the CPC with which the zBX is associated.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/54a9716c-a326-11e0-9469-001f163805d8 HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 24. Get zBX Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:01 GMT
content-type: application/json;charset=UTF-8
content-length: 704
{
   "class": "zbx",
   "current-dpxi50z-entitlements": 0,
   "current-isaopt-entitlements": 0,
   "current-power-entitlements": 4,
   "current-systemx-entitlements": 2,
   "description": "Represents one zBX",
   "is-locked": false,
   "machine-model": "002",
   "machine-serial": "0000000ZBX26",
   "machine-type": "2458",
   "max-dpxi50z-entitlements": 28,
   "max-isaopt-entitlements": 28,
   "max-power-entitlements": 28,
   "max-systemx-entitlements": 28,
   "name": "2458-002-0000000ZBX26",
   "object-id": "54a9716c-a326-11e0-9469-001f163805d8",
   "object-uri": "/api/zbxs/54a9716c-a326-11e0-9469-001f163805d8",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340"
}
```

Figure 25. Get zBX Properties: Response

Inventory service data

Information about the zBXs managed by the HMC can be optionally included in the inventory data provided by the Web Services API Inventory Service.

Inventory entries for zBX objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"zbx"** are to be included. An entry for a particular zBX is included only if the API user has object-access permission to the CPC with which that object is associated.

For each zBX object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for "Get zBX Properties" on page 76. That is, the data provided is the same as would be provided if a **Get zBX Properties** operation were requested targeting this object.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the **Get Inventory** response to describe a single zBX. This object would appear as one array entry in the response array:

```
{
   "class": "zbx",
   "current-dpxi50z-entitlements": 0,
   "current-isaopt-entitlements": 0,
   "current-power-entitlements": 4,
   "current-systemx-entitlements": 2,
   "description": "Represents one zBX",
   "is-locked": false,
   "machine-model": "002",
   "machine-serial": "0000000ZBX26",
   "machine-type": "2458",
   "max-dpxi50z-entitlements": 28,
   "max-isaopt-entitlements": 28,
   "max-power-entitlements": 28,
   "max-systemx-entitlements": 28,
   "name": "2458-002-0000000ZBX26",
   "object-id": "28ba8930-7bc4-11e0-a905-001f163803de",
   "object-uri": "/api/zbxs/28ba8930-7bc4-11e0-a905-001f163803de",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340"
}
```

Figure 26. zBX object: Sample inventory data

zBX Top-of-Rack switches

A Top-of-Rack Switch element of a zBX represents one of the VLAN capable switches provided by the zBX to serve as the network infrastructure of the Intra-Ensemble Data Network (IEDN). (Note that the Top-of-Rack switches used for management purposes are considered internal components of zManager and are not surfaced to API clients as TOR objects.)

Data model

The Top-of-Rack switch object provides the following properties:

Table 19. zBX Top-of-Rack switches: base managed object properties specializations

Name	Qualifier	Type	Description
element-uri	—	String/ URI	The canonical URI path for a TOR object is of the form `/api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}` where *{tor-id}* is the value of the **element-id** property of the Top-of-Rack switch object.
element-id	—	String	The element ID of the TOR.
parent	—	String/ URI	The canonical URI path of the zBX that is the parent of the TOR.
class	—	String	The class of a TOR object is **"top-of-rack-switch"**.
name	—	String (32)	The name of the TOR. This name cannot be changed.
tor-ports-list	—	Array of tor-port-info objects	Array of nested tor-port-info objects (described in the next table), each entry of which describes one port on the TOR switch.

The following tor-port-info nested object describes the properties of a single TOR port:

Table 20. zBX Top-of-Rack switches: tor-port-info nested object properties

Name	Qualifier	Type	Description
element-uri	—	String/ URI	Canonical URI of the port for this TOR: `/api/zbxs/{zbx-id}/top-of-rack-switches/{tor-id}/ports/{port-id}` when *{port-id}* is the value of the **port-num** property of the port.
port-num	—	String (3)	The TOR port number, as a string of 1 to 3 numeric digits.
type	—	String Enum	This is the type of port. TOR ports have the following types: • **"internal"** - Provides connectivity to an ISAOPT coordinator blade. • **"external"** - Provides connectivity to networks external to the IEDN.
port-access	(w)	String Enum	This describes the port access mode associated with this TOR port. Possible values: • **"trunk"** • **"access"**
all-virtual-networks	(w)	Boolean	If true then all defined virtual networks in the ensemble can be used with this TOR port. When true, the **virtual-networks-list** property value will be an empty array. This property can have the value true only if the **type** property has the value **"internal"** and the **port-access-mode** property has the value **"trunk"**. That is, only internal ports configured in trunk mode can be configured to allow access to all defined virtual networks. If true, attempts to invoke the **Add Top-of-Rack Switch Port to Virtual Networks** or **Remove Top-of-Rack Switch Port from Virtual Networks** operations will result in errors. Changing this property from false to true will remove all virtual networks from the current **virtual-networks-list**.
virtual-networks-list	—	Array of URI	This is the list of virtual network URIs that represent the virtual networks that can be used for this port. There can be more than one virtual network URI in this list only if the **port-access** property has the value **"trunk"**. If the **all-virtual-networks** property is false, the **Add Top-of-Rack Switch Port to Virtual Networks** and **Remove Top-of-Rack Switch Port from Virtual Networks** operations may be used to modify this list. Changing the **all-virtual-networks** property from false to true will remove all virtual networks from this list.

Table 20. zBX Top-of-Rack switches: tor-port-info nested object properties (continued)

Name	Qualifier	Type	Description
allow-all-macs	(w)	Boolean	If true, all MAC addresses are allowed to access this port and the **mac-filter-list** array will be empty. If false, only those MAC addresses listed in the **mac-filter-list** property are permitted to access this port. Network traffic from all other MAC addresses is filtered out. If true, attempts to invoke the **Add MAC Filter to Top-of-Rack Switch Port** and **Remove MAC Filter from Top-of-Rack Switch Port** operations will result in errors. Changing the value from false to true will remove all MAC filters from the current **mac-filter-list** property. This property is writable only for TOR ports with a type value of **"external"**. The value of this property is always false for TOR ports with a **type** value of **"internal"**.
mac-filter-list	—	Array of String (17)	The list of MAC addresses to allow (permit) for this port. Each entry in the list is a MAC address. A MAC address is represented as a string of length 17 consisting of 6 groups of two lower-case hexadecimal digits separated by colons (:), e.g. “01:23:45:67:89:ab”. If the **allow-all-macs** property is true, this list is empty. If the **allow-all-macs** property is false, the **Add MAC Filter to Top-of-Rack Switch Port** and **Remove MAC Filter from Top-of-Rack Switch Port** operations may be used to modify this list. Changing the **allow-all-macs** property from false to true will remove all MAC addresses from this list. This property is only applicable to TOR ports with a **type** value of **"external"**. The value of this property is always an empty array of TOR ports with a **type** value of **"internal"**.

Operations

List Top-of-Rack Switches of a zBX

The **List Top-of-Rack Switches of a zBX** operation returns a list of the Top-of-Rack (TOR) switches for a zBX managed by the HMC.

HTTP method and URI

GET /api/zbxs/{*zbx-id*}/top-of-rack-switches

In this request, the URI variable *{zbx-id}* is the object ID of a zBX object for which top-of-rack switches are to be listed.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`tors-list`	Array of objects	Array of nested objects where each element in the array is a TOR-info object, described in the next table.

Each TOR-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/ URI	Canonical URI path of the TOR object.
`element-id`	String	Object ID of the TOR object.
`name`	String	Display name of the TOR object.

Description

On successful completion, this operation obtains a list of URIs that represent the Top-of-Rack (TOR) switches for the zBX designated by *{zbx-id}*. If no TORs are present, then the response body is provided and contains an empty **tors-list** array.

A TOR switch is included in the list only if the API user has object-access permission to the CPC associated with the zBX that contains the TOR switch. If the ensemble contains a TOR switch but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the Ensemble does not contain any TOR switches, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC associated with the zBX containing TOR switches to be listed in the result
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 81.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top of Rack Switch** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
```

Figure 27. List Top-of-Rack Switches: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 23 Nov 2011 20:57:48 GMT
content-type: application/json;charset=UTF-8
content-length: 409
{
   "tors-list": [
      {
         "element-id": "189ad58d-c19f-4cdc-8dc8-639f3ccd4f49",
         "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
       189ad58d-c19f-4cdc-8dc8-639f3ccd4f49",
         "name": "GG0210487100"
      },
      {
         "element-id": "fea63433-e03d-4ea1-a5da-6a2fc7abe844",
         "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
       fea63433-e03d-4ea1-a5da-6a2fc7abe844",
         "name": "GG0210487101"
      }
   ]
}
```

Figure 28. List Top-of-Rack Switches: Response

Get Top-of-Rack Switch Properties

The **Get Top-of-Rack Switch Properties** operation obtains the properties of the TOR specified by the *{tor-id}* managed object.

HTTP method and URI

GET /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the TOR object as defined in the“Data model” on page 79. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

On successful completion, the **Get Top-of-Rack Switch Properties** operation obtains the properties of the TOR specified by *{tor-id}*.

The URI path must designate an existing TOR switch object and the API user must have object-access permission to the CPC associated with the zBX in which that switch resides. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 83.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top of Rack Switch** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844 HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
```

Figure 29. Get Top-of-Rack Switch Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 23 Nov 2011 20:57:53 GMT
content-type: application/json;charset=UTF-8
content-length: 1314
{
   "class": "top-of-rack-switch",
   "element-id": "fea63433-e03d-4ea1-a5da-6a2fc7abe844",
   "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
    fea63433-e03d-4ea1-a5da-6a2fc7abe844",
   "name": "GG0210487101",
   "parent": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3",
   "tor-ports-list": [
      {
         "all-virtual-networks": false,
         "allow-all-macs": true,
         "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
       fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/8",
         "mac-filter-list": [],
         "port-access": "access",
         "port-num": "8",
         "type": "internal",
         "virtual-networks-list": [
            "/api/virtual-networks/4ccb3c0c-a703-11df-a6fc-00215ef9b504"
         ]
      },
      {
         "all-virtual-networks": false,
         "allow-all-macs": true,
         "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
        fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34",
         "mac-filter-list": [],
         "port-access": "trunk",
         "port-num": "34",
         "type": "external",
         "virtual-networks-list": [
            "/api/virtual-networks/4ccb3c0c-a703-11df-a6fc-00215ef9b504"
         ]
      }
   ]
}
```

Figure 30. Get Top-of-Rack Switch Properties: Response

Get Top-of-Rack Switch Port Details

The **Get Top-of-Rack Switch Port Details** operation obtains the properties of the designated TOR port specified by the *{port-id}* identifier.

HTTP method and URI

GET /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the designated TOR port defined in the "Data model" on page 79 by tor-port-info. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

On successful completion, the **Get Top-of-Rack Switch Port Details** operation obtains the properties of the TOR port specified by *{port-id}*.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Manage Virtual Networks** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34 HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
```

Figure 31. Get Top-of-Rack Switch Port Details: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 23 Nov 2011 20:58:19 GMT
content-type: application/json;charset=UTF-8
content-length: 427
{
   "all-virtual-networks": false,
   "allow-all-macs": false,
   "element-uri": "/api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
    fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34",
   "mac-filter-list": [
      "00:24:7e:e0:ea:9e"
   ],
   "port-access": "trunk",
   "port-num": "34",
   "type": "external",
   "virtual-networks-list": [
      "/api/virtual-networks/4ccb3c0c-a703-11df-a6fc-00215ef9b504",
      "/api/virtual-networks/cd42c1c0-1615-11e1-817f-00215e6a0c26"
   ]
}
```

Figure 32. Get Top-of-Rack Switch Port Details: Response

Update Top-of-Rack Switch Port Properties

The **Update Top-of-Rack Switch Port Properties** operation updates selected properties of the specified TOR port.

HTTP method and URI

POST /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Request body contents

The request body is expected to contain a JSON object that provides the new values of any writeable property that is to be updated by this operation. Field names and data types in this JSON object are expected to match the corresponding property names and data types defined by the "Data model" on page 79. The JSON object can and should omit fields for properties whose values are not to be changed by this operation.

Description

This operation updates writeable properties of the TOR port specified by *{port-id}*.

The request body does not need to specify a value for all writeable properties, but rather can and should contain fields for the properties to be updated. Object properties for which no input value is provided remain unchanged by this operation.

The request body is validated against the schema described in the "Request body contents" on page 87. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. In addition to occurring for common validation reasons, status code 400 is returned when the requested changes are not valid considering the port's type, or for inconsistencies in the request and the **port-access-mode** setting.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	133	The port's port-access setting is not valid for the update. A change from **"trunk"** to **"access"** mode is not allowed if more than one virtual network is defined to the port.
	134	The port's type does not support MAC filters. Only ports that have a type of **"external"** support MAC filters.
	139	The port's type does not support the **all-virtual-networks** setting.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top-of-Rack Switch** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34 HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
content-type: application/json
content-length: 49
{
   "allow-all-macs": false,
   "port-access": "trunk"
}
```

Figure 33. Update Top-of-Rack Switch Port Properties: Request

```
204 No Content
date: Wed, 23 Nov 2011 20:57:59 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 34. Update Top-of-Rack Switch Port Properties: Response

Add MAC Filters to Top-of-Rack Switch Port

The **Add MAC Filters to Top-of-Rack Switch Port** operation adds MAC address filters to the designated Top-of-Rack Switch port to permit a list of MAC addresses to connect to this TOR port.

HTTP method and URI

POST /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}/operations/add-mac-filters

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Request body contents

The request body is expected to contain a JSON object that provides the following:

Field name	Type	Rqd/Opt	Description
mac-address-list	Array of String	Required	Array of MAC addresses to add to the **mac-filter-list** for the specified TOR port. If successful, these MAC addresses will be permitted to access the TOR port. This operation will fail if the TOR port's **allow-all-macs** is true.

Description

On successful execution, this operation adds the MAC provided in the **mac-address-list** field to the TOR port's **mac-filter-list**. If the inputs contain addresses that are already in the list, these will be accepted without error. These MAC addresses will be permitted to access the specified TOR port.

The request body is validated against the schema described in "Request body contents" on page 89. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. Consider the following to prevent bad requests:

- Ensure the type of the target port supports the setting of MAC filters
- The **allow-all-macs** property must be false
- Ensure that the MAC address is valid.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	134	The port's type does not support MAC filters. Only ports that have a **type** value of **"external"** support MAC filters.
	135	This operation cannot be performed when the **allow-all-macs** property of the TOR port is true.
	136	The MAC address is invalid.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top-of-Rack-Switch** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34/operations/add-mac-filters HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
content-type: application/json
content-length: 43
{
   "mac-address-list": [
      "00:24:7E:E0:EA:9E"
   ]
}
```

Figure 35. Add MAC Filters to Top-of-Rack Switch Port: Request

```
204 No Content
date: Wed, 23 Nov 2011 20:58:07 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 36. Add MAC Filters to Top-of-Rack Switch Port: Response

Remove MAC Filters from Top-of-Rack Switch Port

The **Remove MAC Filters from Top-of-Rack Switch Port** operation removes MAC address filters from the designated Top-of-Rack Switch port.

HTTP method and URI

POST /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}/operations/remove-mac-filters

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Request body contents

The request body is expected to contain a JSON object that provides the following:

Field name	Type	Rqd/Opt	Description
`mac-address-list`	Array of String	Required	This is an array of the MAC addresses to remove from the MAC mac-filter-list for the specified TOR port.

Description

On successful execution, this operation removes the MAC provided in the **mac-address-list** field to the TOR port's **mac-filter-list**. If a MAC address input is not currently in the **mac-filter-list**, it will be ignored without resulting in an error.

The request body is validated against the schema described in "Request body contents" on page 91. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. Consider the following to prevent bad requests:

- Ensure the type of the target port supports the setting of MAC filters
- The **allow-all-macs** property must be false
- Ensure that the MAC address is valid.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	134	The port's type does not support MAC filters. Only ports that have a **type** value of **"external"** support MAC filters.
	135	This operation cannot be performed when the **allow-all-macs** property of the TOR port is true.
	136	The MAC address is invalid.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top-of-Rack Switches** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34/operations/remove-mac-filters HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
content-type: application/json
content-length: 43
{
   "mac-address-list": [
      "00:24:7E:E0:EA:9E"
   ]
}
```

Figure 37. Remove MAC Filters from Top-of-Rack Switch Port: Request

```
204 No Content
date: Wed, 23 Nov 2011 20:58:40 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 38. Remove Mac Filters from Top-of-Rack Switch Port: Response

Add Top-of-Rack Switch Port to Virtual Networks

The **Add Top-of-Rack Switch Port to Virtual Networks** operation adds Top-of-Rack switch port to the specified list of virtual networks.

HTTP method and URI

POST /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}/operations/add-virtual-networks

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Request body contents

The request body is expected to contain a JSON object that provides the following:

Field name	Type	Rqd/Opt	Description
`add-virtual-networks`	Array of String/ URI	Required	Array of virtual network URIs that define the virtual networks that can be used for the specified TOR port.

Description

On successful execution, this operation adds the specified virtual networks to the TOR port's **virtual-networks-list** array. If the inputs contain virtual networks that are already in the list, these will be accepted without error. Traffic for VLAN IDs representing these virtual networks will now be able to be used on the specified TOR port.

The request body is validated against the schema described in "Request body contents" on page 93. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. Consider the following to prevent bad requests:

- Ensure the **type** of the target port supports the setting of virtual networks
- The target TOR port's **all-virtual-networks** property must be false when issuing this request
- If the port's **port-mode** property is **"access"**, then it only supports one virtual network.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	137	The **port-access** property is **"access"**, therefore, only one virtual network is allowed in the TOR port's **virtual-networks-list**.
	138	This operation cannot be performed when the **all-virtual-networks** property of the TOR port is true.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top-of-Rack Switches** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
	362	A URI in the **remove-virtual-networks** field of the request body does not designate an existing virtual network.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34/operations/add-virtual-networks HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
content-type: application/json
content-length: 88
{
   "add-virtual-networks": [
      "/api/virtual-networks/cd42c1c0-1615-11e1-817f-00215e6a0c26"
   ]
}
```

Figure 39. Add Top-of-Rack Switch Port to Virtual Networks: Request

```
204 No Content
date: Wed, 23 Nov 2011 20:58:18 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 40. Add Top-of-Rack Switch Port to Virtual Networks: Response

Remove Top-of-Rack Switch Port from the Virtual Networks

The **Remove Top-of-Rack Switch Port from the Virtual Networks** operation removes the Top-of-Rack Switch port from the specified virtual networks.

HTTP method and URI

POST /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}/operations/remove-virtual-networks

URI variables:

Variable	Description
{zbx-id}	Object ID of the zBX object containing the TOR switch.
{tor-id}	Element ID of the TOR object.
{port-id}	Element ID of the TOR port.

Request body contents

The request body is expected to contain a JSON object that provides the following:

Field name	Type	Rqd/Opt	Description
`remove-virtual-networks`	Array of String/ URI	Required	List of virtual network URIs to remove from the **virtual-network-list** array for the specified TOR port.

Description

On successful execution, this operation removes the specified virtual networks from the TOR port's **virtual-networks-list** array. If a virtual network in the input is not currently in the **virtual-networks-list**, it will be ignored without resulting in an error.

The request body is validated against the schema described in "Request body contents" on page 95. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. Consider the following to prevent bad requests:

- Ensure the **type** of the target port supports the setting of virtual networks
- The target TOR port's **all-virtual-networks** property must be false when issuing this request.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC associated with the zBX containing the TOR switch
- Action/task permission to **Configure Top-of-Rack Switch** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message:

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	138	This operation cannot be performed when the **all-virtual-networks** property of the TOR port is true.
403 (Forbidden)	0	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Configure Top-of-Rack Switches** task is required.
404 (Not Found)	1	The **object-id** in the URI (*{zbx-id}*) does not designate an existing resource, or designates a resource for which the API user does not have object-access permission.
	360	The TOR element ID in the URI (*{tor-id}*) does not designate an existing top of rack switch.
	361	The port ID the URI (*{port-id}*) does not designate an existing port.
	362	A URI in the **remove-virtual-networks** field of the request body does not designate an existing virtual network.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/zbxs/da5d7720-a337-11e0-9555-00262df332b3/top-of-rack-switches/
  fea63433-e03d-4ea1-a5da-6a2fc7abe844/ports/34/operations/remove-virtual-networks HTTP/1.1
x-api-session: sqernk0lqu0s49oul0drld6ifcb1cplf902og86d9apesqjsl
content-type: application/json
content-length: 91
{
   "remove-virtual-networks": [
      "/api/virtual-networks/cd42c1c0-1615-11e1-817f-00215e6a0c26"
   ]
}
```

Figure 41. Remove Top-of-Rack Switch Port from the Virtual Networks: Request

```
204 No Content
date: Wed, 23 Nov 2011 20:58:32 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 42. Remove Top-of-Rack Switch Port from the Virtual Networks: Response

Rack object

A Rack object represents a Rack that houses the zBX components.

There is at least one Rack object for each zBX, designated as the rack in location B. This rack houses the top of rack (TOR) switches for the zBX as well as one or two BladeCenter chassis. Rack B is the first rack to be populated when installing a zBX. Additional racks are added for larger configurations that contain more than two BladeCenter chassis in the zBX. The additional racks are designated with consecutive location codes (C, D, etc.).

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, but does not provide the operational-status-related properties defined in that schema because it does not maintain the concept of an operational status.

The following class-specific specializations apply to the other Base Managed Object properties:

Table 21. Rack object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
name	(ro)	String (1-64)	The name of the object. Currently, this is assigned by zManager based on the rack's location (has the same value as the location property).[1]
description	—	String	This field is not provided.
object-uri	—	String/ URI	The canonical URI path for a Rack object is of the form `/api/racks/{rack-id}`.
parent	—	String/ URI	The canonical URI path of the parent zBX object, of the form `/api/zbxs/{zbx-id}`.
class	—	String	The value **"rack"**.

Table 21. Rack object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
Note: 1. This **name** property is currently assigned based on the location of the rack and thus has the same value as the **location** property. However, it is possible that the API could be extended to allow this property to be writeable, in which case an API or User-Interface user could change the name to contain arbitrary data. Therefore, API client applications that are interested in determining the location of the rack should not rely on the contents and format of the **name** property, but rather obtain location information from the location property.			

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 22. Rack object: class specific properties

Name	Type	Description
`serial-number`	String	The 12 character serial number of the rack.
`location`	String	The zManager assigned location code for the rack, as an uppercase alphabetic character starting with "B" for the first rack of the zBX and with subsequent racks being designated by consecutive letters.

Operations

List Racks of a zBX

The **List Racks of a zBX** operation lists the racks where the zBX components are mounted.

HTTP method and URI

`GET /api/zbxs/{zbx-id}/racks`

In this request, the URI variable *{zbx-id}* is the object ID of the zBX object whose racks are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`racks`	Array of objects	Array of nested rack-info objects, described in the next table

Each nested rack-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the rack object in the form /api/racks/*{rack-id}*
`name`	String	The **name** property of the rack object

Description

The **List Racks of a zBX** operation lists the racks where the zBX components are mounted. BladeCenters are plugged into the rack, and the blades are plugged into the BladeCenters. The object URI and name are provided for each rack.

If the **name** query parameter is specified, then a rack is included in the list only if the name pattern matches the **name** property of the object.

A rack is included in the list only if the API user has object-access permission for the associated CPC. If the HMC is a manager of a zBX, but the API user does not have permission to the CPC with which the zBX is associated, that object is omitted from the list, but no error status code results.

If the HMC does not manage any racks, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC with which the zBX and rack objects are associated.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 98.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{zbx-id}* does not designate a zBX object, or the API user does not have object access permission to the CPC with which it is associated.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/54a9716c-a326-11e0-9469-001f163805d8/racks HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 43. List Racks of a zBX: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:01 GMT
content-type: application/json;charset=UTF-8
content-length: 142
{
   "racks": [
      {
         "name": "B",
         "object-uri": "/api/racks/b434398a-a328-11e0-9b4a-001f163805d8"
      }
   ]
}
```

Figure 44. List Racks of a zBX: Response

Get Rack Properties

The **Get Rack Properties** operation retrieves the properties of a single rack object that is designated by its object ID.

HTTP method and URI

GET /api/racks/{*rack-id*}

In this request, the URI variable *{rack-id}* is the object ID of the rack object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the rack object as defined in the "Data model" on page 97. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get zBX Properties** operation returns the current properties for the rack object specified by *{rack-id}*.

On successful execution, all of the current properties as defined in "Data model" on page 97 for the rack object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing rack object and the API user must have object-access permission to the CPC with which it is associated. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC with which the rack object designated by **{rack-id}** is associated.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{rack-id}* does not designate an existing rack object, or the API user does not have object access permission to the CPC with which the rack is associated.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/racks/04419e1e-a9db-11e0-8077-f0def1610d20 HTTP/1.1
x-api-session: 5wksmtktms30ajeohh0bn411fdzusmklld4jyddd1des5t78aq
```

Figure 45. Get Rack Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 14:22:52 GMT
content-type: application/json;charset=UTF-8
content-length: 272
{
   "class": "rack",
   "description": "Rack",
   "is-locked": false,
   "location": "C",
   "name": "C",
   "object-id": "04419e1e-a9db-11e0-8077-f0def1610d20",
   "object-uri": "/api/racks/04419e1e-a9db-11e0-8077-f0def1610d20",
   "parent": "/api/zbxs/291e385e-a9cd-11e0-8650-f0def1610d20",
   "serial-number": "12345"
}
```

Figure 46. Get Rack Properties: Response

Inventory service data

Information about the zBXs racks managed by the HMC can be optionally included in the inventory data provided by the Web Services API Inventory Service.

Inventory entries for Rack objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"rack"** are to be included. An entry for a particular Rack is included only if the API user has object-access permission to the CPC with which that object is associated.

For each Rack object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for "Get Rack Properties" on page 100. That is, the data provided is the same as would be provided if a **Get Rack Properties** operation were requested targeting this object.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the **Get Inventory** response to describe a single Rack. This object would appear as one array entry in the response array:

```
{
   "class": "rack",
   "description": "Rack",
   "is-locked": false,
   "location": "B",
   "name": "B",
   "object-id": "28bc03c8-7bc4-11e0-a905-001f163803de",
   "object-uri": "/api/racks/28bc03c8-7bc4-11e0-a905-001f163803de",
   "parent": "/api/zbxs/28ba8930-7bc4-11e0-a905-001f163803de",
   "serial-number": "123456123456"
}
```

Figure 47. Rack object: Sample inventory data

BladeCenter object

A BladeCenter object represents a single BladeCenter of the zBX.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, including the operational-status properties, with the following class-specific specialization:

Table 23. BladeCenter object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
name	(ro)	String (1-64)	The zManager-assigned name of the zBX BladeCenter.[1]
description	—	String	This property is not provided.
object-uri	—	String/URI	The canonical URI path for a zBX BladeCenter object is of the form `/api/bladecenters/{bladecenter-id}`
parent	—	String/URI	The canonical URI path of the parent Rack object, of the form `/api/racks/{rack-id}`
class	—	String	The value **"bladecenter"**
status	(sc)	String Enum	The status of the blade center. Values: • **"no-power"** • **"operating"**
additional-status	—	String Enum	This property is not provided.

Note:

1. This **name** property is currently assigned based on the location of the BladeCenter and is of the form **RackName.BladecenterName** (e.g. B.1). However, it is possible that the API could be extended to allow this property to be writeable, in which case an API or User-Interface user could change the name to contain arbitrary data. Therefore, API client applications that are interested in determining the location of the BladeCenter should not rely on the contents and format of the **name** property, but rather obtain location information from the **location** property.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 24. BladeCenter object: class specific properties

Name	Qualifier	Type	Description
machine-type	—	String	4 characters
machine-model	—	String	3 characters
machine-serial	—	String	7 characters
location	—	String (4)	4 Characters (RxxB) RxxB – BladeCenter vertical position in rack R
has-hardware-messages	(pc)	Boolean	The BladeCenter has a hardware message (true) or the BladeCenter does not have a hardware message (false).

Energy Management Related Additional Properties

In addition to the properties defined above, this object includes the following additional class-specific properties related to energy management. For further explanation of the various states involved, please see the "Overview" section in the "Energy Management" chapter of this document.

Table 25. BladeCenter object: energy management related additional properties

Name	Qualifier	Type	Description
power-rating	—	Integer	Specifies the maximum power usage of this BladeCenter in watts (W). This value is a calculated value, as indicated by the electrical rating labels or system rating plates of the BladeCenter components.
power-consumption	(mg)	Integer	Specifies the current power consumption in watts (W) of this BladeCenter. The BladeCenter power consumption includes the power consumption of the Blades contained within the BladeCenter and the shared infrastructure.
power-saving	—	String Enum	Specifies the current power saving setting of the BladeCenter. Power saving reduces the energy consumption of a system, and can be managed it using the Set Power Saving operation. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption and performance of the BladeCenter. This is the default setting. • **"low-power"** - Specifies low power consumption for all components of the BladeCenter enabled for power saving. • **"custom"** - Specifies that some, but not all, components of the BladeCenter are in the Low power setting. • **"not-supported"** - Specifies that power saving is not supported for this BladeCenter. • **"not-available"** - Specifies that **power-saving** property could not be read from this BladeCenter. • **"not-entitled'** - Specifies that the server is not entitled to power saving.

Table 25. BladeCenter object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
power-saving-state	—	String Enum	Specifies the power saving setting of the BladeCenter set by the user. Note that this property indicates the user setting and may not match the real state of the hardware compared to the **power-saving** property. For more information, see “Group power saving” on page 139. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption and performance of the BladeCenter. This setting will be applied to all children. • **"low-power"** - Specifies low power consumption for all components of the BladeCenter enabled for power saving. • **"custom"** - Specifies that the BladeCenter does not control the children. This is the default setting. • **"not-supported"** - Specifies that power saving is not supported for this BladeCenter. • **"not-entitled'** - Specifies that the server is not entitled to power saving.
power-save-allowed	—	String Enum	Should be used to determine if a call of the power save operation is currently allowed. If a value other that "allowed" is returned the caller may reckon that the power save operation will fail. The possible settings include: • **"allowed"** - Alter power save setting is allowed for this BladeCenter • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power saving. • **"not-supported"** - Specifies that power saving is not supported for this BladeCenter. • **"under-group-control"** - The BladeCenter is under group control and cannot be individually altered.
power-capping-state	—	String Enum	Specifies the current power capping setting of the BladeCenter. Power capping limits peak power consumption of a system, and you can manage it by using the Set Power Cap operation. The possible settings include: • **"disabled"** - Specifies not setting the power cap of the BladeCenter and not limiting the peak power consumption. This is the default setting. • **"enabled"** - Specifies capping all components of the BladeCenter available for power capping to limit the peak power consumption of the BladeCenter. • **"custom"** - Specifies individually configuring the components of the BladeCenter for power capping. • **"not-supported"** - Specifies that power capping is not supported for this BladeCenter. • **"not-entitled"** - Specifies that the server is not entitled to power capping.
power-cap-minimum	—	Integer	Specifies the minimum value for the BladeCenter cap value in watts (W). This is a sum of the component minimum cap values.
power-cap-maximum	—	Integer	Specifies the maximum value for the BladeCenter cap value in watts (W). This is a sum of the component maximum cap values.

Table 25. BladeCenter object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
power-cap-current	—	Integer	Specifies the current cap value for the BladeCenter in watts (W). The current cap value indicates the power budget for the BladeCenter and is the sum of the component Cap values.
power-cap-allowed	—	String Enum	Should be used to determine if a call of the power capping operation is currently allowed. If a value other that **"allowed"** is returned the caller may reckon that the power capping operation will fail. The possible settings include: • **"allowed"** - Alter power capping setting is allowed for this BladeCenter • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power capping. • **"not-supported"** - Specifies that power capping is not supported for this BladeCenter. • **"under-group-control"** - The BladeCenter is under group control and cannot be individually altered.
ambient-temperature	(mg)	Float	Specifies the input air temperature in degrees Celsius (ºC) as measured by the system.
exhaust-temperature	—	Float	Specifies the exhaust air temperature in degrees Celsius (ºC) as calculated by the system. This is useful in determining potential hot spots in the data center.

Operations

List BladeCenters in a Rack

The **List BladeCenters in a Rack** operation lists the BladeCenters that are mounted into the rack.

HTTP method and URI

`GET /api/racks/{rack-id}/bladecenters`

In this request, the URI variable *{rack-id}* is the object ID of the rack object whose BladeCenters are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`blade-centers`	Array of objects	Array of nested BladeCenter-info objects, described in the next table. If the rack does not have any BladeCenters mounted in it, an empty array is provided.

Each nested BladeCenter-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the BladeCenter object in the form /api/bladecenters/*{bladecenter-id}*
`name`	String	The **name** property of the BladeCenter object (for example, B.1)
`status`	String Enum	The **status** property of the BladeCenter object

Description

The **List BladeCenters in a Rack** operation lists the BladeCenters that are mounted in the rack. The object URI, name, and status are provided for each BladeCenter.

If the **name** query parameter is specified, then a BladeCenter is included in the list only if the name pattern matches the **name** property of the object.

A BladeCenter is included in the list only if the API user has object-access permission for that object. If the HMC is a manager of a BladeCenter, but the API user does not have permission to it, that object is omitted from the list, but no error status code results.

If the HMC does not manage any BladeCenters, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC with which the rack specified by the URI is associated
- Object access permission to any BladeCenter object are to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 105.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{rack-id}* does not designate a rack object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/racks/04419e1e-a9db-11e0-8077-f0def1610d20/bladecenters HTTP/1.1
x-api-session: 5wksmtktms30ajeohh0bn411fdzusmklld4jyddd1des5t78aq
```

Figure 48. List BladeCenters in a Rack: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 14:22:52 GMT
content-type: application/json;charset=UTF-8
content-length: 231
{
   "blade-centers": [
      {
         "name": "C.1",
         "object-uri": "/api/bladecenters/22e79848-3957-35dc-b88e-c661f9c8b680",
         "status": "operating"
      },
      {
         "name": "C.2",
         "object-uri": "/api/bladecenters/e3ee0adc-27c0-355e-93b9-ace8a3d2da15",
         "status": "operating"
      }
   ]
}
```

Figure 49. List BladeCenters in a Rack: Response

List BladeCenters in a zBX

The **List BladeCenters in a zBX** operation lists the BladeCenters in a zBX.

HTTP method and URI

GET /api/zbxs/{*zbx-id*}/bladecenters

In this request, the URI variable *{zbx-id}* is the object ID of the zBX object whose BladeCenters are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`blade-centers`	Array of objects	Array of nested BladeCenter-info objects, described in the next table. If the zBX does not have any BladeCenters, an empty array is provided.

Each nested BladeCenter-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the BladeCenter object in the form /api/bladecenters/*{bladecenter-id}*
`name`	String	The **name** property of the BladeCenter object (for example, B.1)
`status`	String Enum	The **status** property of the BladeCenter object

Description

The **List BladeCenters in a zBX** operation lists the BladeCenters in the zBX. The object URI, name, and status are provided for each BladeCenter.

If the **name** query parameter is specified, then a BladeCenter is included in the list only if the name pattern matches the **name** property of the object.

A BladeCenter is included in the list only if the API user has object-access permission for that object. If the HMC is a manager of a BladeCenter, but the API user does not have permission to it, that object is omitted from the list, but no error status code results.

If the HMC does not manage any BladeCenters, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC with which the zBX specified by the URI is associated
- Object access permission to any BladeCenter object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 107.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{zbx-id}* does not designate a zBX object, or the API user does not have object access permission to the CPC with which it is associated.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/291e385e-a9cd-11e0-8650-f0def1610d20/bladecenters HTTP/1.1
x-api-session: 5wksmtktms30ajeohh0bn411fdzusmklld4jyddd1des5t78aq
```

Figure 50. List BladeCenters in a zBX: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 14:22:52 GMT
content-type: application/json;charset=UTF-8
content-length: 443
{
   "blade-centers": [
      {
         "name": "C.1",
         "object-uri": "/api/bladecenters/22e79848-3957-35dc-b88e-c661f9c8b680",
         "status": "operating"
      },
      {
         "name": "B.1",
         "object-uri": "/api/bladecenters/2ae200b3-fa8e-3db7-b34a-ec08780aaac6",
         "status": "operating"
      },
      {
         "name": "C.2",
         "object-uri": "/api/bladecenters/e3ee0adc-27c0-355e-93b9-ace8a3d2da15",
         "status": "operating"
      },
      {
         "name": "B.2",
         "object-uri": "/api/bladecenters/f5bca837-bc0d-34e6-bcef-766411287439",
         "status": "operating"
      }
   ]
}
```

Figure 51. List BladeCenters in a zBX: Response

Get BladeCenter Properties

The **Get BladeCenter Properties** operation retrieves the properties of a single BladeCenter object that is designated by its object ID.

HTTP method and URI

GET /api/bladecenters/{*bladecenter-id*}

In this request, the URI variable *{bladecenter-id}* is the object ID of the BladeCenter object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the rack object as defined in the "Data model" on page 102. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get BladeCenter Properties** operation returns the current properties for the BladeCenter object specified by *{bladecenter-id}*.

On successful execution, all of the current properties as defined in "Data model" on page 102 for the BladeCenter object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing BladeCenter object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the BladeCenter object specified by *{bladecenter-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 109.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{bladecenter-id}* does not designate an existing BladeCenter object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 52. Get BladeCenter Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:02 GMT
content-type: application/json;charset=UTF-8
content-length: 948
{
   "acceptable-status": [
      "operating"
   ],
   "ambient-temperature": 18.0,
   "class": "bladecenter",
   "description": "Represents one BladeCenter",
   "exhaust-temperature": 25.5,
   "has-hardware-messages": false,
   "has-unacceptable-status": false,
   "is-locked": false,
   "location": "B01B",
   "machine-model": "HC1",
   "machine-serial": "KQZZXLF",
   "machine-type": "8852",
   "name": "B.2",
   "object-id": "ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "object-uri": "/api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "parent": "/api/racks/b434398a-a328-11e0-9b4a-001f163805d8",
   "power-cap-allowed": "allowed",
   "power-cap-current": 9561,
   "power-cap-maximum": 9561,
   "power-cap-minimum": 3473,
   "power-capping-state": "custom",
   "power-consumption": 1876,
   "power-rating": 9561,
   "power-save-allowed": "allowed",
   "power-saving": "high-performance",
   "status": "operating"
}
```

Figure 53. Get BladeCenter Properties: Response

Inventory service data

Information about the BladeCenter chassis managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for BladeCenter objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"bladecenter"** are to be included. An entry for a particular BladeCenter is included only if the API user has object-access permission to that object.

For each BladeCenter object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for "Get BladeCenter Properties" on page 109. That is, the data provided is the same as would be provided if a **Get BladeCenter Properties** operation were requested targeting this object.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the **Get Inventory** response to describe a single BladeCenter. This object would appear as one array entry in the response

array:

```
{
   "acceptable-status": [
      "operating"
   ],
   "additional-status": "unknown",
   "ambient-temperature": 18.5,
   "class": "bladecenter",
   "description": "Represents one BladeCenter",
   "exhaust-temperature": 24.5,
   "has-hardware-messages": false,
   "has-unacceptable-status": false,
   "is-locked": false,
   "location": "B10B",
   "machine-model": "HC1",
   "machine-serial": "KQZZXLF",
   "machine-type": "8852",
   "name": "B.1",
   "object-id": "ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "object-uri": "/api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "parent": "/api/racks/28bc03c8-7bc4-11e0-a905-001f163803de",
   "power-cap-allowed": "under-group-control",
   "power-cap-current": 9444,
   "power-cap-maximum": 9444,
   "power-cap-minimum": 4042,
   "power-capping-state": "disabled",
   "power-consumption": 1875,
   "power-rating": 9444,
   "power-save-allowed": "allowed",
   "power-saving": "high-performance",
   "status": "operating"
}
```

Figure 54. BladeCenter object: Sample inventory data

Blade object

A blade object represents a single blade in the zBX.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, including the operational-status properties, with the following class-specific specialization:

Table 26. Blade object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
name	(ro)	String (1-64)	The zManager assigned name of the blade[1, 2]
description	—	String	This property is not provided.
object-uri	—	String/URI	The canonical URI path for a zBX Blade object is of the form `/api/blade/{blade-id}`.
parent	—	String/URI	The canonical URI path for a parent BladeCenter object, in the form `/api/bladecenter/{bladecenter-id}`.[2]
class	—	String	The value **"blade"**.

Table 26. Blade object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
status	(sc)	String Enum	The status of the Blade object. Possible values: • **"no-power"** - the blade is powered off • **"status-check"** - the blade and the console are not communicating • **"not-operating"** - the blade is powered on and communicating with the console but is not running work • **"stopped"** - operations on the blade are quiesced • **"definition-error"** - an error has occurred when loading the blade with the firmware • **"operating"** - blade is operating normally.
additional-status	—	String Enum	This property is not provided.

Notes:

1. This **name** property is currently assigned based on the location of the Blade and is of the form **RackName.BladecenterName.BladeSlot** (e.g. B.1.01). However, it is possible that the API could be extended to allow this property to be writeable, in which case an API or User-Interface user could change the name to contain arbitrary data. Therefore, API client applications that are interested in determining the location of the blade should not rely on the contents and format of the **name** property, but rather obtain location information from the **location** property.
2. The location of a blade can be moved from slot to slot within a zBX. When a blade is moved to a different slot, the original URI of this blade is retained. Because the blade **name**, **parent** and **location** is based on the slot location of the blade, these three properties can change for a given URI when the blade is moved within the zBX. The relocation of a blade generates an inventory change notification to report the removal of the blade, then an inventory change notification to report the addition of the blade. Upon addition of the blade, expect the values of these properties to differ.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 27. Blade object: class specific properties

Name	Qualifier	Type	Description	Supported "type" values
type	—	String Enum	Type of the blade. Values: • **"power"** – the System z Power® blade • **"system-x"** – the System x blade • **"isaopt"** – the IBM Smart Analytics Optimizer blade • **"dpxi50z"** – the DataPower® XI50 blade.	All
processors	—	Integer	number of zBX blade processors.	All
cores-per-processor	—	Integer	The number of processing cores provided by each processor of the zBX blade.	All
memory-size	—	Integer	memory size of the zBX blade specified in MB.	All
machine-type	—	String	4 characters.	All
machine-model	—	String	3 characters.	All
machine-serial	—	String	12 characters.	All
location	—	String	8 Characters (RxxBBSyy)[1] • RxxB – BladeCenter vertical position in rack R • BSyy – physical slot of Blade Server	All

Table 27. Blade object: class specific properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
isaopt-mode	(pc)	String Enum	The mode of the ISAOPT blade. Values: • **"worker"** • **"coordinator"**	isaopt
virtualization-host	—	String/ URI	The canonical URI path for the virtualization host being hosted by the blade.	power, system-x
has-hardware-messages	(pc)	Boolean	The blade has a hardware message (true) or the blade does not have a hardware message (false).	All
licensed-features	—	String	The Features that this DPXI50Z blade is licensed for. Each licensed feature in the string is delimited with commas (,). The blade must be Operating to retrieve this property; if it is not, null is returned instead.	dpxi50z
iedn-interfaces	(pc)	Array of iedn-interface Objects	Complex object defining the IEDN Interfaces configured to this DPXI50Z blade. The blade must be Operating to retrieve this property; if it is not, null is returned instead.	dpxi50z
Note: [1]The location of a blade can be moved from slot to slot within a zBX. When a blade is moved to a different slot, the original URI of this blade is retained. Because the blade **name**, **parent** and **location** is based on the slot location of the blade, these three properties can change for a given URI when the blade is moved within the zBX. The relocation of a blade generates an inventory change notification to report the removal of the blade, then an inventory change notification to report the addition of the blade. Upon addition of the blade, expect the values of these properties to differ.				

Energy management related additional properties: In addition to the properties defined above, this object includes the following additional class-specific properties related to energy management. For further explanation of the various states involved, see Chapter 9, "Energy management," on page 137.

Table 28. Blade object: energy management related additional properties

Name	Qualifier	Type	Description
power-rating	—	Integer	Specifies the maximum power usage in watts (W) of this blade. This is a calculated value, as indicated by the electrical rating label or system rating plate of the blade.
power-consumption	(mg)	Integer	Specifies the current power consumption in watts (W) for this blade.
power-saving	—	String Enum	Specifies the current power saving setting of the blade. Power saving is used to reduce the energy consumption of a system and can be managed in the Set Power Saving operation. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption of the blade. This is the default setting. • **"low-power"** - Specifies reducing the performance of the blade to allow for low power consumption. • **"not-supported"** - Specifies power saving is not supported for this blade. • **"not-available"** - Specifies **power-saving** property could not be read from this blade. • **"not-entitled"** - Specifies the server is not entitled to power saving. Additional power savings modes may be introduced as zManager is extended to support additional power saving capabilities.

Table 28. Blade object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
power-saving-state	—	String Enum	Specifies the power saving setting of the Blade set by the user. Please note that this property indicates the user setting and may not match the real state of the hardware compared to the **power-saving** property. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption of the blade. This is the default setting. • **"low-power"** - Specifies low power consumption for all components of the blade enabled for power saving. • **"not-supported"** - Specifies power saving is not supported for this blade. • **"not-entitled"** - Specifies the server is not entitled to power saving.
power-save-allowed	—	String Enum	Should be used to determine if a call of the power save operation is currently allowed. If a value other than **"allowed"** is returned the caller may reckon that the power save operation will fail. The possible settings include: • **"allowed"** - Alter power save setting is allowed for this blade • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power saving. • **"not-supported"** - Specifies power saving is not supported for this blade. • **"under-group-control"** - The blade is under group control and cannot be individually altered.
power-capping-state	—	String Enum	Specifies the current power capping setting of the blade. Power capping limits peak power consumption of a system, and you can manage it with the Set Power Cap operation. The possible settings include: • **"disabled"** - Specifies not setting the power cap of the blade and not limiting the peak power consumption. This is the default setting. • **"enabled"** - Specifies limiting the peak power consumption of the blade to the current cap value. • **"not-supported"** - Specifies that power capping is not supported for this blade. • **"not-entitled"** - Specifies that the server is not entitled to power capping.
power-cap-minimum	—	Integer	Specifies the minimum value for the blade cap value in watts (W).
power-cap-maximum	—	Integer	Specifies the maximum value for the blade cap value in watts (W).
power-cap-current	—	Integer	Specifies the current cap value for the blade in watts (W). The current cap value indicates the power budget for the blade.

Table 28. Blade object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
power-cap-allowed	—	String Enum	Should be used to determine if a call of the power capping operation is currently allowed. If a value other than **"allowed"** is returned the caller may reckon that the power capping operation will fail. The possible settings include: • **"allowed"** - Alter power capping setting is allowed for this blade • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power capping. • **"not-supported"** - Specifies power capping is not supported for this blade. • **"under-group-control"** - The blade is under group control and cannot be individually altered.

IEDN interface nested object: The IEDN (Intraensemble Data network) Interface object defines the VLan sub-interface on the DPXI50Z blade:

Table 29. Blade object: IEDN interface nested object properties

Name	Type	Description
name	String (1-64)	Display name of the IEDN Interface. Valid characters are: a->z, A->Z, 0->9, underscore, dash and period. Periods can not be consecutive.
is-active	Boolean	Administrative State of the IEDN Interface configuration. Values: • True – Enabled means that the Op-State of the IEDN interface will be up if there are no errors in the configuration. • False – Disabled means that the Op-State of the IEDN interface will always be down.
network-uri	String/URI	The canonical URI of the virtual network to which this IEDN interface is connected. The combination of the virtual network with the Ethernet Interface must be unique per blade. That is, each interface is associated with one virtual network, and the virtual network associated with one interface cannot also be associated with another.
ethernet-interface	String Enum	The physical Ethernet network interface, either **"eth7"** or **"eth9"**. The combination of the virtual network with the Ethernet Interface must be unique per blade.
ip-address	String	IP Address in either IPv4 or IPv6 format. No default value is provided.
net-mask	String	Network Mask associated with the IP Address in either IPv4 or IPv6 format. If an ipv4 ip-address is provided, valid values are 0-32, the default is 32. If an ipv6 ip-address is provided, valid values are 0-128, the default is 128.
secondary-ip-address	Array of Strings	List of secondary IP Addresses of either IPv4 or IPv6 format. No default value is provided.

Table 29. Blade object: IEDN interface nested object properties (continued)

Name	Type	Description
secondary-net-mask	Array of Strings	List of secondary Network Masks associated with the Secondary IP Addresses of either IPv4 or IPv6 format. If a secondary ipv4 ip-address is provided, valid values are 0-32, the default is 32. If a secondary ipv6 ip-address is provided, valid values are 0-128, the default is 128.
ipv4-gateway	String	The IPv4 address to use for the default IPv4 gateway. No default value is provided.
ipv6-gateway	String	The IPv6 address to use for the default IPv6 gateway. No default value is provided.
is-ipv6-auto-config-enabled	Boolean	True if the IPv6 Auto configuration is enabled. Otherwise, false. When enabled, the interface is configured with a link-local secondary address. When disabled, uses the defined primary address. Default is false.
is-slaac	Boolean	IPv6 Auto configuration must be enabled to utilize this option. True if IPv6 Stateless Address is enabled. Otherwise, false. When enabled, the IPv6 address is obtained when connected to the network. When disabled, the interface uses the defined primary address. Default is false.
dad-transmit	Integer	IPv6 Auto configuration must be enabled to utilize this option. Specify the number of duplicate address detection (DAD) attempts to perform. Default is 1.
dad-transmit-delay	Integer	IPv6 Auto configuration must be enabled to utilize this option. When the number of duplicate address detection (DAD) attempts is greater than 1, specify the delay between attempts in milliseconds. Default is 1000.
mac-address	String (17)	The MAC address represented as 6 groups of two lower-case hexadecimal digits separated by colons, e.g. “01:23:45:67:89:ab”. Length is 17 characters. The MAC address uses the ensemble prefix.

Operations

List Blades in a BladeCenter

The **List Blades in a BladeCenter** operation lists all the blades in a BladeCenter.

HTTP method and URI

GET /api/bladecenters/{*bladecenter-id*}/blades

In this request, the URI variable *{bladecenter-id}* is the object ID of the BladeCenter object whose blades are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`blades`	Array of objects	Array of nested blade-info objects, described in the next table. If the BladeCenter does not have any blades mounted in it, an empty array is provided.

Each nested blade-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the blade object in the form /api/blades/*{blade-id}*
`name`	String	The **name** property of the blade object (for example, B.1.01)
`status`	String Enum	The **status** property of the blade object
`type`	String Enum	The type of the blade.

Description

The **List Blades in a BladeCenter** operation lists the blades that are in the BladeCenter. The object URI, name, status, and type are provided for each blade.

If the **name** query parameter is specified, then a blade is included in the list only if the name pattern matches the **name** property of the object.

A blade is included in the list only if the API user has object-access permission for that object. If the HMC is a manager of a blade, but the API user does not have permission to it, that object is omitted from the list, but no error status code results.

If the HMC does not manage any blades, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission the BladeCenter object specified by the URI
- Object access permission to any blade object are to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{bladecenter-id}* does not designate a BladeCenter object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
GET /api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E/blades HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 55. List Blades in a BladeCenter: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:02 GMT
content-type: application/json;charset=UTF-8
content-length: 386
{
   "blades": [
      {
         "name": "B.2.02",
         "object-uri": "/api/blades/938706AC3FF111D78B5600215EC0330E",
         "status": "operating",
         "type": "power"
      },
      {
         "name": "B.2.03",
         "object-uri": "/api/blades/B8210BC02D1E11E0AE81E41F13FE1430",
         "status": "operating",
         "type": "system-x"
      }
   ]
}
```

Figure 56. List Blades in a BladeCenter: Response

List Blades in a zBX

The **List Blades in a zBX** operation lists all the blades in all the BladeCenters in a zBX. This operation has an optional parameter to specify the blade type to return in the list. If this parameter is omitted, all blades of all blade types are returned.

HTTP method and URI

GET /api/zbxs/{zbx-id}/blades

In this request, the URI variable *{zbx-id}* is the object ID of the zBX object whose blades are to be obtained.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) used to limit returned objects based on matching the objects **name** property
type	String	Optional	Filter string used to limit returned objects to those that have a matching **type** property. Value must be a valid blade **type** property value. To request that the results include blades of multiple types, specify this parameter multiple times. For example "type=power&type=isaopt".

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`blades`	Array of objects	Array of nested blade-info objects, described the next table. If the zBX does not have any blades, an empty array is provided.

Each nested blade-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the blade object in the form /api/blades/*{blade-id}*
`name`	String	The **name** property of the blade object (for example, B.1.01)
`status`	String Enum	The **status** property of the blade object
`type`	String Enum	The type of the blade

Description

The **List Blades in a zBX** operation lists the blades in the zBX. The object URI, name, status, and type are provided for each blade.

A blade is included in the list only if the API user has object-access permission for that object. If the HMC is a manager of a blade, but the API user does not have permission to it, that object is omitted from the list, but no error status code results.

If the **name** query parameter is specified, the returned list is limited to those blades that have a **name** property matching at least one specified name filter pattern. If the **name** parameter is omitted, this filtering is not done.

If the **type** query parameter is specified, the parameter is validated to ensure it is a valid blade **type** property value. If the value is not valid, a 400 (Bad Request) is returned. If the value is valid, the returned list is limited to those blades that have a **type** property matching a specified type value. If the **type** parameter is omitted, this filtering is not done.

If both the **name** and **type** query parameters are specified, a blade is included in the list only if it passes both the name and type filtering criteria.

If the HMC does not manage any blades, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC with which the zBX specified by the URI is associated
- Object access permission to any blade object are to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 120.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{zbx-id}* does not designate a zBX object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/zbxs/54a9716c-a326-11e0-9469-001f163805d8/blades HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 57. List Blades in a zBX: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:02 GMT
content-type: application/json;charset=UTF-8
content-length: 386
{
   "blades": [
      {
         "name": "B.2.02",
         "object-uri": "/api/blades/938706AC3FF111D78B5600215EC0330E",
         "status": "operating",
         "type": "power"
      },
      {
         "name": "B.2.03",
         "object-uri": "/api/blades/B8210BC02D1E11E0AE81E41F13FE1430",
         "status": "operating",
         "type": "system-x"
      }
   ]
}
```

Figure 58. List Blades in a zBX: Response

Get Blade Properties

The **Get Blade Properties** operation retrieves the properties of a single blade object that is designated by its object ID.

HTTP method and URI

GET /api/blades/{*blade-id*}

In this request, the URI variable *{blade-id}* is the object ID of the blade object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the rack object as defined in the “Data model” on page 102. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get Blade Properties** operation returns the current properties for the blade object specified by *{blade-id}*.

On successful execution, all of the current properties as defined in“Data model” on page 112 for the blade object are provided in the response body, and HTTP status code 200 (OK) is returned. If the blade is a DPXI50z blade and its current status is not **"operating"**, then null is returned as the value of the **licensed-features** and **iedn-interfaces** properties, but the operation otherwise succeeds.

The URI path must designate an existing blade object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the blade object specified by *{blade-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 122.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{blade-id}* does not designate an existing blade object, the object ID designates a blade object that is not of the correct type, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/blades/B8210BC02D1E11E0AE81E41F13FE1430 HTTP/1.1
x-api-session: 5q07hx6jgp2ngn2cypq1zxot76sfwnzky0ih8nddd5hz6bpiue
```

Figure 59. Get Blade Properties: Request

```
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 21 Jul 2011 17:49:03 GMT
content-type: application/json;charset=UTF-8
content-length: 989
{
   "acceptable-status": [
      "operating"
   ],
   "class": "blade",
   "cores-per-processor": 8,
   "has-hardware-messages": false,
   "has-unacceptable-status": false,
   "is-locked": false,
   "location": "B01BBS03",
   "machine-model": "AC1",
   "machine-serial": "06NL721",
   "machine-type": "7872",
   "memory-size": 131072,
   "name": "B.2.03",
   "object-id": "B8210BC02D1E11E0AE81E41F13FE1430",
   "object-uri": "/api/blades/B8210BC02D1E11E0AE81E41F13FE1430",
   "parent": "/api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "power-cap-allowed": "allowed",
   "power-cap-current": 268,
   "power-cap-maximum": 500,
   "power-cap-minimum": 268,
   "power-capping-state": "enabled",
   "power-consumption": 241,
   "power-rating": 500,
   "power-save-allowed": "unknown",
   "power-saving": "not-supported",
   "processors": 2,
   "status": "operating",
   "type": "system-x",
   "virtualization-host": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d"
}
```

Figure 60. Get Blade Properties: Response for blade of type "system-x" (similar for type "power")

```
200 OK
x-request-id: Sx3 Rx13
x-client-correlator: 21
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 14:22:55 GMT
content-type: application/json;charset=UTF-8
content-length: 899
{
   "acceptable-status": [
      "operating"
   ],
   "class": "blade",
   "cores-per-processor": 8,
   "has-hardware-messages": false,
   "has-unacceptable-status": false,
   "iedn-interfaces": [],
   "is-locked": false,
   "licensed-features": " MQ, TAM, DataGlue, JAXP-API, PKCS7-SMIME, SQL-ODBC,
     WebSphere-JMS, RaidVolume, iSCSI, LocateLED, AppOpt, zBX",
   "location": "C01BBS01",
   "machine-model": "4BX",
   "machine-serial": "6800442",
   "machine-type": "4195",
   "memory-size": 12288,
   "name": "C.2.01",
   "object-id": "eadb0be8-6fdb-11df-8f6a-e41f137a29e4",
   "object-uri": "/api/blades/eadb0be8-6fdb-11df-8f6a-e41f137a29e4",
   "parent": "/api/bladecenters/e3ee0adc-27c0-355e-93b9-ace8a3d2da15",
   "power-cap-allowed": "under-group-control",
   "power-cap-current": 444,
   "power-cap-maximum": 444,
   "power-cap-minimum": 144,
   "power-capping-state": "disabled",
   "power-consumption": 115,
   "power-rating": 444,
   "power-save-allowed": "unknown",
   "power-saving": "not-supported",
   "processors": 2,
   "status": "operating",
   "type": "dpxi50z"
}
```

Figure 61. Get Blade Properties: Response for blade of type "dpx150z":

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 14:22:55 GMT
content-type: application/json;charset=UTF-8
content-length: 773
{
   "acceptable-status": [
      "operating"
   ],
   "class": "blade",
   "cores-per-processor": 8,
   "has-hardware-messages": false,
   "has-unacceptable-status": false,
   "is-locked": false,
   "isaopt-mode": "coordinator",
   "location": "B10BBS01",
   "machine-model": "PEL",
   "machine-serial": "KQWZTNG",
   "machine-type": "7870",
   "memory-size": 49152,
   "name": "B.1.01",
   "object-id": "fa8d1eea-95ab-33cf-bf86-a03cc1346222",
   "object-uri": "/api/blades/fa8d1eea-95ab-33cf-bf86-a03cc1346222",
   "parent": "/api/bladecenters/2ae200b3-fa8e-3db7-b34a-ec08780aaac6",
   "power-cap-allowed": "under-group-control",
   "power-cap-current": 515,
   "power-cap-maximum": 500,
   "power-cap-minimum": 220,
   "power-capping-state": "disabled",
   "power-consumption": 181,
   "power-rating": 500,
   "power-save-allowed": "unknown",
   "power-saving": "not-supported",
   "processors": 2,
   "status": "operating",
   "type": "isaopt"
}
```

Figure 62. Get Blade Properties: Response for blade of type "isaopt":

Activate a Blade

The **Activate a Blade** operation activates a blade object designated by its object ID. This operation is asynchronous.

HTTP method and URI

POST /api/blades/{*blade-id*}/operations/activate

In this request, the URI variable *{blade-id}* is the object ID of the blade object to activate.

Response body contents

Once the activation request is accepted, the response body is a JSON object with the following fields:

Field name	Type	Description
job-uri	String/ URI	URI that may be queried to retrieve activation status updates.

Asynchronous result description

Once the activation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Activate a Blade** operation.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. (For more information, see "Query Job Status" on page 44.) When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) that are set. (See the description that follows.) The **job-results** field is null for asynchronous blade activation jobs.

Description

The **Activate a Blade** operation activates the blade object specified by *{blade-id}*. Activation brings the blade into a state of "operating". If the blade is already powered on when the activation operation is requested, the blade is powered off and then brought to a state of "operating". If the blade is a host to a virtualization application, then this application is activated also. See the "Activating a Virtualization Host" on page 202 for more information.

The URI path must designate an existing blade object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Activate** task; otherwise, status code 409 (Conflict) is returned.

When the blade activation job is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the activation job. See "Query Job Status" on page 44 for information on how to query job status. When the activate job has completed, an asynchronous result message is sent with job status and reason codes described in "Job status and reason codes" on page 128.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the blade object specified by *{blade-id}*
- Action/task permission to the **Activate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 126.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID *{blade-id}* does not designate an existing blade object, or the API user does not have object access permission to it.
409 (Conflict)	1	The object ID *{blade-id}* designates a blade object that is not in the correct state (status) for performing the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status code	Reason code	Description
200 (OK)	N/A	Activation completed successfully.
500 (Server Error)	100	Blade activation failed.
500 (Server Error)	101	Blade activation job timed out.

Example HTTP interaction

```
POST /api/blades/62f508a6-2d21-11e0-813b-e41f13fe15a8/operations/activate HTTP/1.1
x-api-session: 6c6s3b1p02v9x9az6brcic9q9dk34jjhxwllw0sqegu0ktia5k
```

Figure 63. Activate a Blade: Request

```
202 Accepted
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 05:45:45 GMT
content-type: application/json;charset=UTF-8
content-length: 60
{
   "job-uri": "/api/jobs/b8824300-1728-11e1-aea4-0010184c8334"
}
```

Figure 64. Activate a Blade: Response

Deactivate a Blade

The **Deactivate a Blade** operation deactivates a blade object designated by its object ID. This operation is asynchronous.

HTTP method and URI

POST /api/blades/{*blade-id*}/operations/deactivate

In this request, the URI variable *{blade-id}* is the object ID of the blade object to deactivate.

Response body contents

Once the activation request is accepted, the response body is a JSON object with the following fields:

Field name	Type	Description
job-uri	String/ URI	URI that may be queried to retrieve deactivation status updates.

Asynchronous result description

Once the deactivation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Deactivate a Blade** operation.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. (For more information, see "Query Job Status" on page 44). When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) that are set. (See the description that follows.) The **job-results** field is null for asynchronous blade activation jobs.

Description

The **Deactivate a Blade** operation activates the blade object specified by *{blade-id}*. Deactivation powers off the blade after an orderly shutdown of any hardware and software activity running on the blade. If the blade is a host to a virtualization application, then this application is deactivated also. See the "Deactivating a Virtualization Host" on page 203 for more information.

The URI path must designate an existing blade object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Deactivate** task; otherwise, status code 403 (Forbidden) is returned. If the blade is not in the correct state to perform the deactivate, a status code of 409 (Conflict) is returned.

When the blade deactivation job is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the deactivation job. See "Query Job Status" on page 44 for information on how to query job status. When the activate job has completed, an asynchronous result message is sent with job status and reason codes described in "Job status and reason codes" on page 130.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the blade object specified by *{blade-id}*
- Action/task permission to the **Deactivate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 128.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID *{blade-id}* does not designate an existing blade object, or the API user does not have object access permission to it.
409 (Conflict)	1	The object ID *{blade-id}* designates a blade object that is not in the correct state (status) for performing the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status code	Reason code	Description
200 (OK)	N/A	Deactivation completed successfully.
500 (Server Error)	100	Blade deactivation failed.
500 (Server Error)	101	Blade deactivation job timed out.

Example HTTP interaction

```
POST /api/blades/62f508a6-2d21-11e0-813b-e41f13fe15a8/operations/deactivate HTTP/1.1
x-api-session: 6c6s3b1p02v9x9az6brcic9q9dk34jjhxwllw0sqegu0ktia5k
```

Figure 65. Deactivate a Blade: Request

```
202 Accepted
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 05:45:04 GMT
content-type: application/json;charset=UTF-8
content-length: 60
{
   "job-uri": "/api/jobs/a0247864-1728-11e1-aea4-0010184c8334"
}
```

Figure 66. Deactivate a Blade: Response

Create IEDN Interface for a DataPower XI50z Blade

The **Create IEDN Interface for a DataPower XI50z Blade** operation adds an IEDN interface with the designated properties to the DataPower XI50z blade configuration.

HTTP method and URI

POST /api/blades/{*blade-id*}/iedn-interface

In this request, the URI variable *{blade-id}* is the object ID of the DPXI50Z blade object to which the IEDN interface is to be added.

Request body contents

The request body contains the following writeable IEDN interface properties of the DPXI50Z blade object that will be used to create the IEDN interface:

Field name	Type	Rqd/Opt	Description
name	String (1-64)	Required	The name of the IEDN interface. Valid characters are: a-z, A-Z, 0-9, underscore, dash and period. Periods cannot be consecutive. Must be unique per blade.

Field name	Type	Rqd/Opt	Description
is-active	Boolean	Required	Administrative state of the IEDN interface configuration. Values: • True – Enabled means that the Op-State of the IEDN interface will be up if there are no errors in the configuration • False – Disabled means that Op-State of the IEDN interface will always be down
network-uri	String/URI	Required	The canonical URI of the virtual network to which this IEDN interface is connected. The combination of the virtual network with the Ethernet interface must be unique per blade. That is, each interface is associated with one virtual network, and the virtual network associated with one interface cannot be associated with another.
ethernet-interface	String Enum	Required	The physical Ethernet network interface – either **"eth7"** or **"eth9"**. The combination of the virtual network with the Ethernet interface must be unique per blade.
ip-address	String	Optional	An IP address in either IPv4 or IPv6 format. No default value is provided.
net-mask	String	Optional	The network mask associated with the IP address in either IPv4 or IPv6 format. If an IPv4 **ip-address** is provided, valid values are 0-32; the default is 32. If an IPv6 **ip-address** is provided, valid values are 0-128; the default is 128.
secondary-ip-address	List of Strings	Optional	A list of secondary IP addresses in either IPv4 or IPv6 format. No default value is provided.
secondary-net-mask	List of Strings	Optional	A list of secondary network masks associated with the secondary IP addresses in either IPv4 or IPv6 format. If a secondary IPv4 **ip-address** is provided, valid values are 0-32, the default is 32. If a secondary IPv6 **ip-address** is provided, valid values are 0-128, the default is 128
ipv4-gateway	String	Optional	The IPv4 address to use for the default IPv4 gateway. No default value is provided.
ipv6-gateway	String	Optional	The IPv6 address to use for the default IPv6 gateway No default value is provided.
is-ipv6-auto-config-enabled	Boolean	Optional	True if the IPv6 auto configuration is enabled. Otherwise, false. When enabled, the interface is configured with a link-local secondary address. When disabled, the interface uses the defined primary address. Default is false.

Field name	Type	Rqd/Opt	Description
is-slaac	Boolean	Optional	IPv6 auto configuration must be enabled to utilize this option. True if IPv6 stateless address is enabled. Otherwise, false. When enabled, the IPv6 address is obtained when connected to the network. When disabled, the interface uses the defined primary address. Default is false.
dad-transmit	Integer	Optional	IPv6 auto configuration must be enabled to utilize this option. The number of duplicate address detection (DAD) attempts to perform. Default is 1.
dad-transmit-delay	Integer	Optional	IPv6 auto configuration must be enabled to utilize this option. When the number of duplicate address detection (DAD) attempts is greater than 1, the delay between attempts in milliseconds. Default is 1000.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
element-uri	String/ URI	Canonical URI path of the IEDN interface object, in the form /api/blade/*{blade-id}*/iedn-interfaces/*{iedn-interface-id}*, where *{iedn-interface-id}* is the value of the **name** property.

Description

The **Create IEDN Interface for a DataPower XI50z Blade** operation creates the IEDN interface for the DataPower XI50z blade as specified by the given properties. The DPXI50Z blade must be operating to perform this operation. Any properties identified as optional may be excluded from the request body. If an optional property is not found in the request body, its value will be set to its default value.

The add of the IEDN interface is permitted if the API user has object-access permission for that DataPower XI50z blade.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the DPXI50z blade object specified by *{blade-id}*
- Action/task permission to the **zBX Blade Details** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents"

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have task access authority to the **zBX Blade Details** task.
404 (Not Found)	1	The object ID *{blade-id}* does not designate an existing blade object, or the API user does not have object access permission to it.
409 (Conflict)	1	The object ID *{blade-id}* designates a blade object that is not in the correct state (status) for performing the requested operation. The blade must be operating to perform this operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Delete IEDN Interface for a DataPower XI50z Blade

The **Delete IEDN Interface for a DataPower XI50z Blade** operation removes the specified IEDN interface for a DataPower XI50z blade.

HTTP method and URI

DELETE /api/blades/{*blade-id*}/iedn-interface/{*iedn-interface-id*}

URI variables

Variable	Description
{iedn-interface-id}	Element ID of the IEDN interface
{blade-id}	Object ID of the blade

Description

The **Delete IEDN Interface for a DataPower XI50z Blade** operation deletes the IEDN interface that is defined for the DataPower XI50z blade. The DPXI50Z blade must be operating to perform this operation.

The IEDN interface is removed only if the API user has object-access permission for that DataPower XI50z blade.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the DPXI50z blade object specified by *{blade-id}*
- Action/task permission to the **zBX Blade Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID *{blade-id}* does not designate an existing blade object, or the API user does not have object access permission to it.
	2	The object ID *{iedn-interface-id}* does not exist on the HMC.
404 (Not Found)	1	The object ID *{blade-id}* designates a blade object that is not in the correct state (status) for performing the requested operation. The blade must be operating to perform this operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Inventory service data

Information about the lades managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for blade objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by inventory class, implicitly via a containing category, or by default) that objects of the various blade type-specific inventory classes are to be included. An entry for a particular blade is included only if the API user has object-access permission to that blade and the applicable type-specific inventory class has been specified, as described in the following table:

Inventory class	Includes blades with "type" value
power-blade	power
system-x-blade	system-x
dpz150z-blade	dpz150z
isaopt-blade	isaopt

For each blade object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for "Get Blade Properties" on page 122. That is, the data provided is the same as would be provided if a **Get Blade Properties** operation were requested targeting this object.

Sample inventory data

The following fragments are examples of the JSON objects that would be included in the **Get Inventory** response to describe a single blade object of a particular type. These objects would appear as array entries in the response array.

```
{
   "acceptable-status": [
      "operating"
   ],
   "class": "blade",
   "cores-per-processor": 8,
   "has-hardware-messages": false,
   "has-unacceptable-status": true,
   "is-locked": false,
   "location": "B10BBS13",
   "machine-model": "71Y",
   "machine-serial": "06C9FDA",
   "machine-type": "8406",
   "memory-size": 32768,
   "name": "B.1.13",
   "object-uri": "/api/blades/D5C5CB8A3F5511D78B5600215EC03866",
   "parent": "/api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "power-cap-allowed": "under-group-control",
   "power-cap-current": 277,
   "power-cap-maximum": 382,
   "power-cap-minimum": 277,
   "power-capping-state": "disabled",
   "power-consumption": 151,
   "power-rating": 382,
   "power-save-allowed": "under-group-control",
   "power-saving": "high-performance",
   "processors": 8,
   "status": "status-check",
   "type": "power",
   "virtualization-host": "/api/virtualization-hosts/baa17718-2990-11e0-8d5b-001f163803de"
}
```


Figure 67. Activate a Blade: Sample inventory data for a blade of type "power"

```
{
   "acceptable-status": [
      "operating"
   ],
   "class": "blade",
   "cores-per-processor": 8,
   "has-hardware-messages": false,
   "has-unacceptable-status": true,
   "is-locked": false,
   "location": "B10BBS12",
   "machine-model": "AC1",
   "machine-serial": "06NL728",
   "machine-type": "7872",
   "memory-size": 131072,
   "name": "B.1.12",
   "object-uri": "/api/blades/62F508A62D2111E0813BE41F13FE15A8",
   "parent": "/api/bladecenters/ECEC1940F05B39EA9B3AEA5C1600AB1E",
   "power-cap-allowed": "under-group-control",
   "power-cap-current": 278,
   "power-cap-maximum": 519,
   "power-cap-minimum": 278,
   "power-capping-state": "disabled",
   "power-consumption": 249,
   "power-rating": 519,
   "power-save-allowed": "unknown",
   "power-saving": "not-supported",
   "processors": 2,
   "status": "status-check",
   "type": "system-x",
   "virtualization-host": "/api/virtualization-hosts/47d3a864-82e1-11e0-b9e4-f0def10bff8d"
}
```

Figure 68. Activate a Blade: Sample inventory data for a blade of type "system-x"

Chapter 9. Energy management

Energy Management is a management task that is pervasive and spread across several components in the systems management stack. Each layer in the stack needs to implement two key functions:

- A set of management functions appropriate for this level at the stack. Energy management functions provided by lower layers can be used to implement these functions.
- Management interfaces are provided that allows management layers above to configure and control the energy management functions.

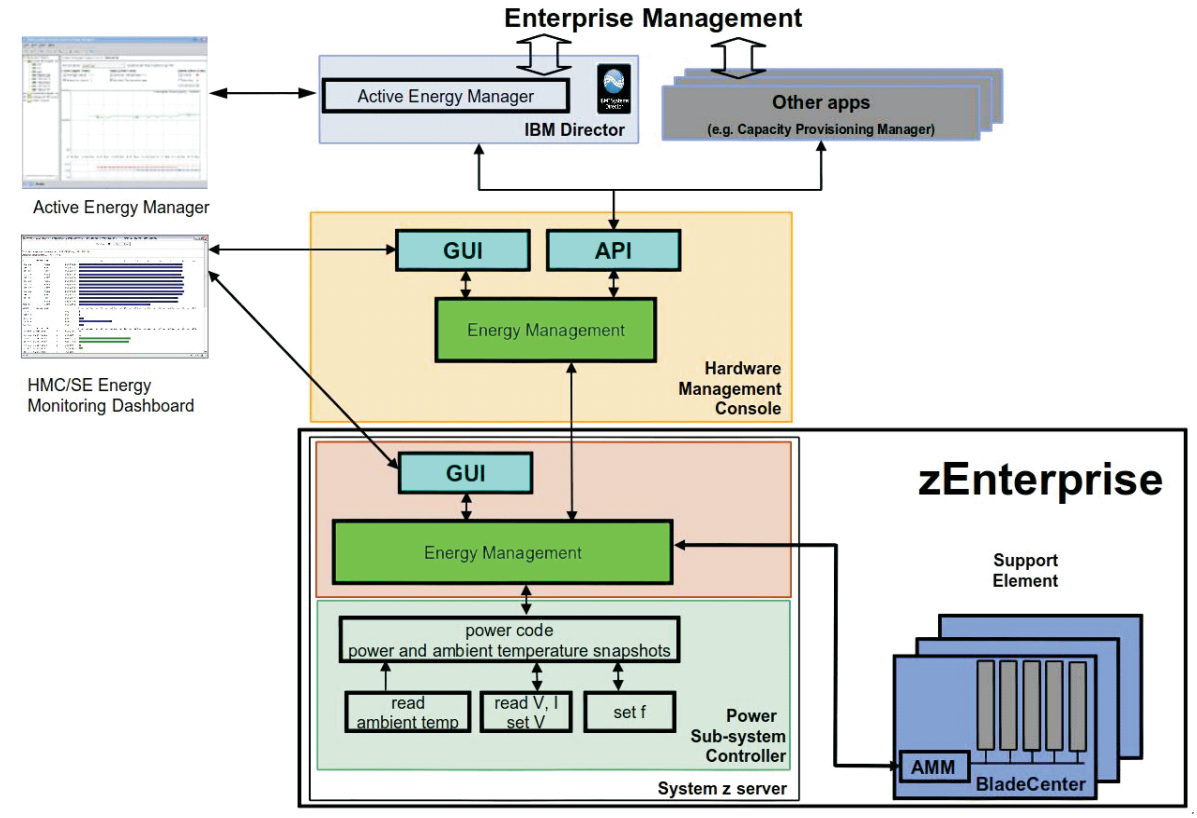

Figure 69. Energy management as applied throughout layers of enterprise management

To achieve this several pieces are needed:

Power and thermal monitoring

"You can't improve what you don't measure" is a trivial engineering paradigm. Measuring energy consumption and the thermal environment is key for management. Energy Monitoring for System z was initially introduced with z9®. Starting with zEnterprise 196 the power consumption of attached BladeCenters is made available and factored into the presented system level power consumption.

Energy control

Based on the measurement data - either for an individual system or aggregated for a group of servers or even a complete data center - analytics can be implemented. These can keep a watch on given limits or can identify optimizing potentials. At a system level energy control mechanisms will be provided to allow for changing energy consumption of a system. These energy controls can be categorized into two groups:

- **Power saving** - Power saving mechanisms are used to reduce the average energy consumption of a system. Through powering off components or reducing performance the power saving is typically achieved. For zEnterprise 196 the Static Power Savings Mode is implemented that reduces processor frequency and voltage for power saving purposes.
- **Power capping** - Power capping is a means to limit peak power consumption of a system. This is especially important in constraint data center environments. Today power and cooling allocation in data centers is usually done via the label power. This typically leads to a significant overprovisioning. Through power capping the power allocation for a system can be adjusted better to the real power consumption of a system and therefore more servers can be deployed within the same physical limits of their data center.

Groups

A group is composed of an object that contains groups or another object and the object or objects it contains. For example, a group might be a CPC that contains a zCPC and optional BladeCenters. Another group might be a BladeCenter containing blades. Please note that only the predefined groups CPC and BladeCenter exist as shown in the following figure:

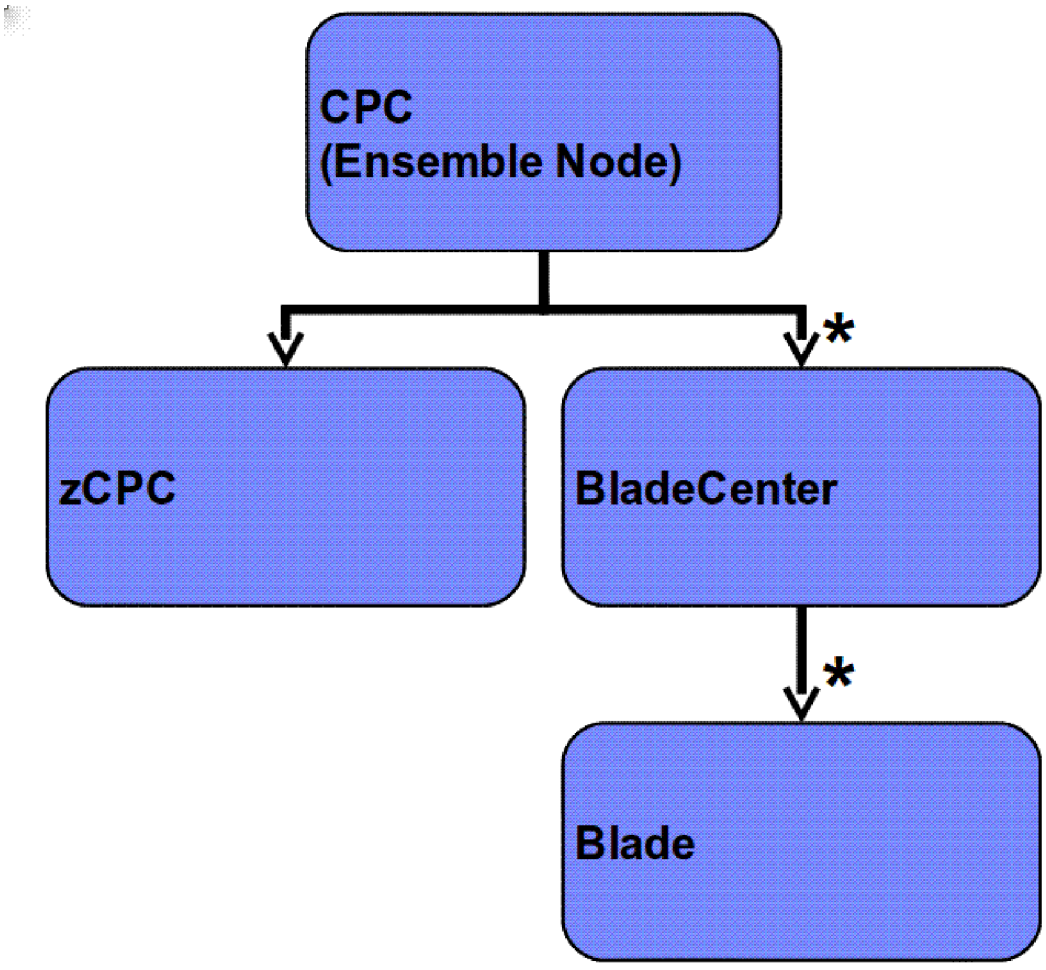

Figure 70. Example of a group and the objects it contains

Special states

In this chapter, the following states are used but the reasoning behind the states isn't always clear. So they are explained here in more detail:

"custom"
Occurs only on groups and indicates that the group does not control the children. Clients are able to alter the children of a group individually.

"under-group-control"
Occurs only on children and indicates that a group controls the state. When clients want to alter the state, the group must be set to **"custom"** first.

"not-supported"
Indicates that the feature (either power saving or power capping) is currently not supported, possible reasons can be:

- Hardware does not support it → permanent
- Firmware level does not support it → can change after a firmware update
- The hardware is not powered on → can change after the device is powered on.

"not-available"
Couldn't read the state of the underlying hardware.

"not-entitled"
Indicates that the automate feature is not installed and so power saving and power capping is not allowed.

Power saving

Power saving is a function that reduces the energy consumption of a system. Please note that power saving is only available if the Automate management enablement feature is installed. The possible settings include:

High performance
The power consumption and performance of the object are not reduced. This is the default setting.

Low power
The performance of the object is reduced to allow for low power consumption. When this setting is selected for CPC and BladeCenter objects, all components of the object enabled for power saving have reduced performance to allow for low power consumption. Use this setting to enable group power saving.

Note: You can only set the power saving setting of the zCPC to Low power one time per calendar day in an air cooled system. This power save property is set to Not Supported if the current zCPC power saving setting is High performance but the zCPC has already entered Low power once within the calendar day.

Custom
Use Custom to disable group power saving and individually configure the components of the object for power saving.

Note: This setting is available only for CPC and BladeCenter objects.

Group power saving

The following are important concepts regarding group power saving:

- Group power saving settings replace individual object settings--that is, the Power Saving setting of a CPC or BladeCenter supersede the Power Saving setting of any object contained within the CPC or BladeCenter.
- You can enable group power saving by setting the Power Saving setting of the CPC or BladeCenter to Low power or High performance.
- You can change individual Power Saving settings only if the object is not under group power saving control.

- To disable group power saving without changing the individual Power Saving settings of the group members, change the Power Saving setting of the CPC or BladeCenter to Custom.

Power capping

Please note that power capping is only available if the Automate management enablement feature is installed.

Group capping

The following are important concepts regarding group power capping:

- Group caps replace individual object caps—that is, the Cap Value of a CPC or BladeCenter supersede the power cap of any object contained within the CPC or BladeCenter.
- You can enable group capping by setting the Power Capping setting of the CPC or BladeCenter to Enabled.
- You can change individual Cap Values if the object is not under group capping control.
- If a CPC or BladeCenter contains an object that does not support power capping, the Power Rating is used in calculating the minimum power cap value for the group. The Power Rating can be found on the details window for an object.
- The maximum Cap Value for a group is the sum of the Power Rating of all group objects.
- When a group component is powered off or removed, the group cap is redistributed to the remaining group components.
- To disable group capping without changing the individual power caps of the group members, change the Power Capping setting of the CPC or BladeCenter to Custom.

Energy management operations summary

The following tables provide an overview of the operations provided. All POST operation were executed asynchronous and provide a query URI where the result of the request can be queried.

Note: The zCPC is not modeled as a full entity like CPC, BladeCenters or blades, because only energy management needs the zCPC to represent only System z hardware without zBX. That is the reason why all zCPC related operations are tied to the CPC.

Table 30. Energy management: operations summary

Operation name	HTTP method and URI path
"Set CPC Power Save" on page 141	`POST /api/cpcs/{cpc-id}/operations/set-cpc-power-save`
"Set CPC Power Capping" on page 143	`POST /api/cpcs/{cpc-id}/operations/set-cpc-power-capping`
"Get CPC Energy Management Data" on page 149	`GET /api/cpcs/{cpc-id}/energy-management-data`
"Set zCPC Power Save" on page 146	`POST /api/cpcs/{cpc-id}/operations/set-zcpc-power-save`
"Set zCPC Power Capping" on page 147	`POST /api/cpcs/{cpc-id}/operations/set-zcpc-power-capping`
"Set BladeCenter Power Save" on page 151	`POST /api/bladecenters/{bladecenter-id}/operations/set-power-save`
"Set BladeCenter Power Capping" on page 153	`POST /api/bladecenters/{bladecenter-id}/operations/set-power-capping`
"Set Blade Power Save" on page 156	`POST /api/blades/{blade-id}/operations/set-power-save`
"Set Blade Power Capping" on page 158	`POST /api/blades/{blade-id}/operations/set-power-capping`

Table 31. Energy management: URI variables

Variable	Description
{cpc-id}	Object ID of a CPC
{bladecenter-id}	Object ID of a BladeCenter
{blade-id}	Object ID of a blade

Energy management for CPC object

The energy management for the CPC object represents all energy management for the CPC.

Data model

The data model for a CPC object includes some properties related to energy management. These properties are described in "Energy management related additional properties" on page 519.

Set CPC Power Save

Use the **Set CPC Power Save** operation to set the power save setting of a CPC.

HTTP method and URI

POST /api/cpc/{*cpc-id*}/operations/set-cpc-power-save

In this request, the URI variable *{cpc-id}* is the object ID of the CPC.

Request body contents

The request body is a JSON object with the following fields:

Name	Type	Rqd/Opt	Description
`power-saving`	Enum string	Required	The possible settings are: • **"high-performance"** - The power consumption and performance of the CPC are not reduced. This is the default setting. • **"low-power"** - Low power consumption for all components of the CPC enabled for power saving. • **"custom"** - Components may have their own settings changed individually. No component settings are actually changed when this mode is entered.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to control the average energy consumption of a CPC object designated by *{cpc-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated CPC is under group control (see "Group capping" on page 140) or the **cpc-power-saving** property of the CPC is set to **"not-supported"** or **"not-entitled"**. (See "Energy management related additional properties" on page 519 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-saving settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described in "HTTP status and reason codes." After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Save** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 141.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully

Job status codes	Job reason code	Description
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the blade or System z hardware
	162	Communication error occurred while trying to access the blade or System z hardware
	163	An error occurred at one or more children

If the job reason code is 163, the **job-results** field provided by the **Query Job Status** operation will contain an object with the following fields:

Field name	Type	Description
`errors`	Object array	A list of error objects, containing detailed error information about errors occurred on children
`at-least-one-operation-succeed`	Boolean	**True** indicates that the operation was successful for at least one child.

Each error object has this structure:

Job status codes	Job reason code	Description
`object-uri`	String URI	The canonical URI path for a specific object where the error occurred
`reason-code`	Integer	Specify the specific error type, possible values are: • 160 - A firmware error occurred while executing the operation • 161 - A hardware error occurred while performing the energy management operation • 162 - Communication error occurred while trying to access the hardware
`message`	String	A non-localized message provided for development purposes only. Client applications should not display this message directly to the user.

Set CPC Power Capping

Use the **Set CPC Power Capping** operation to set the power capping settings of a CPC.

HTTP method and URI

`POST /api/cpcs/{cpc-id}/operations/set-cpc-power-capping`

In this request, the URI variable *{cpc-id}* is the object ID of the CPC.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`power-capping-state`	Enum string	Required	The possible settings are: • **"disabled"** - The power cap of the CPC is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - The peak power consumption of the CPC is limited to the current cap value. • **"custom"** - Individually configure the components of the BladeCenter for power capping. No component settings are actually changed when this mode is entered.
`power-cap-current`	Integer	Optional	Specifies the current cap value for the CPC in watts (W). The current cap value indicates the power budget for the CPC. This field is only required if the **power-capping-state** field is set to **"enabled"**. The **power-cap-current** must be between **cpc-power-cap-minimum** and **cpc-power-cap-maximum**: **cpc-power-cap-minimum** <= value <= **cpc-power-cap-maximum**

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to limit the peak power consumption of a CPC object designated by *{cpc-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated CPC is under group control (see "Group capping" on page 140) or the **power-capping-state** property of the CPC is set to **"not-supported"** or **"not-entitled"**. (See "Energy management related additional properties" on page 519 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-capping settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described in "HTTP status and reason codes" on page 145. After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Capping** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 144.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
400 (Bad Request)	7	The **power-cap-current** field contains a value that is not in the range **cpc-power-cap-minimum** ... **cpc-power-cap-maximum**
	5	The **power-cap-current** field is not set, but **power-capping-state** field is set to **"enabled"**.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the blade or System z hardware
	162	Communication error occurred while trying to access the blade or System z hardware
	163	An error occurred at one or more children

If the job reason code is 163, the **job-results** field provided by the **Query Job Status** operation will contain an object with the following fields:

Field name	Type	Description
`errors`	Object array	A list of error objects, containing detailed error information about errors occurred on children
`at-least-one-operation-succeed`	Boolean	**True** indicates that the operation was successful for at least one child.

Each error object has this structure:

Job status codes	Job reason code	Description
`object-uri`	String URI	The canonical URI path for a specific object where the error occurred
`reason-code`	Integer	Specify the specific error type, possible values are: • 160 - A firmware error occurred while executing the operation • 161 - A hardware error occurred while performing the energy management operation • 162 - Communication error occurred while trying to access the hardware
`message`	String	A non localized message provided for development purposes only. Client applications should not display this message directly to the user.

Set zCPC Power Save

Use the **Set zCPC Power Save** operation to set the power save settings of the zCPC portion of a CPC.

HTTP method and URI

`POST /api/cpc/{cpc-id}/operations/set-zcpc-power-save`

In this request, the URI variable *{cpc-id}* is the object ID of the CPC.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`power-saving`	Enum string	Required	The possible settings are: • **"high-performance"** - The power consumption and performance of the zCPC are not reduced. This is the default setting. • **"low-power"** - Low power consumption for all components of the zCPC enabled for power saving.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to control the average energy consumption of a zCPC portion of the CPC *{cpc-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated zCPC is under group control (see “Group capping” on page 140) or the **zcpc-power-saving** property of the zCPC is set to **"not-supported"** or **"not-entitled"**. (See “Energy management related additional properties” on page 519 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-saving settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body

includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described in "HTTP status and reason codes." After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Save** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in"Response body contents" on page 146.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the System z hardware
	162	Communication error occurred while trying to access the System z hardware

Set zCPC Power Capping

Use the **Set zCPC Power Capping** operation to set the power capping settings of the zCPC portion of a CPC.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/set-zcpc-power-capping

In this request, the URI variable *{cpc-id}* is the object ID of the CPC.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
power-capping-state	Enum string	Required	The possible settings are: • **"disabled"** - The power cap of the zCPC is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - The peak power consumption of the zCPC is limited to the current cap value.
power-cap-current	Integer	Optional	Specifies the current cap value for the zCPC in watts (W). The current cap value indicates the power budget for the zCPC. This field is only required if the **power-capping-state** field is set to **"enabled"**. The **power-cap-current** must be between **zcpc-power-cap-minimum** and **zcpc-power-cap-maximum**: **zcpc-power-cap-minimum** <= value <= **zcpc-power-cap-maximum**

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
job-uri	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to limit the peak power consumption of a zCPC object designated by *{cpc-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated zCPC is under group control (see "Group capping" on page 140) or the **power-capping-state** property of the zCPC is set to **"not-supported"** or **"not-entitled"**. (See "Energy management related additional properties" on page 519 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-capping settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described below. After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to all blade, BladeCenter, CPC and zCPC objects

- Action/task permission to the **Power Capping** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in"Response body contents" on page 148.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
400 (Bad Request)	7	The **power-cap-current** field contains a value that is not in the range **zcpc-power-cap-minimum** ... **zcpc-power-cap-maximum**
	5	The **power-cap-current** field is not set, but **power-capping-state** field is set to **"enabled"**.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the System z hardware
	162	Communication error occurred while trying to access the System z hardware

Get CPC Energy Management Data

Use the **Get CPC Energy Management Data** operation to retrieve all energy management related data in one single call.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/energy-management-data

In this request, the URI variable *{cpc-id}* is the object ID of the CPC.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
objects	Array of objects	An array of nested em-data objects containing the energy management data. The format of each nested object is given in the next table.

Each nested em-data object contains the following fields:

Field name	Type	Description
object-uri	String/ URI	The canonical URI path of the specific object to which this em-data object pertains.
object-id	String	Object-id property of the specific object to which this em-data object pertains.
class	String	The type of the specific object to which this em-data object pertains.
properties	Object	Nested object containing the energy management properties for the object identified by the object-uri field, as described in the data model section for objects of the type indicated by the class field.
error-occurred	Boolean	If true, indicates that an error occurred while querying the data for the object specified by the object-uri. As a consequence the property could be null or incomplete.

Description

The **Get CPC Energy Management Data** is a convenience operation to allow a client to reprieve all energy management related data for a CPC in a single request rather than invoking several requests to retrieve this data.

Note that this operation returns data for a child object of the designated CPC only if the API user has object-access permission to that object. Children objects for which the API user does not have access are omitted from the response and no error is indicated.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC object designated by the request, and for any children objects for which data is to be returned.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See"Common request validation reason codes" on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Energy management for BladeCenter object

Data model

The data model for a BladeCenter object includes some properties related to energy management. These properties are described in Chapter 8, "zBX infrastructure elements," on page 69, under the data model for the BladeCenter object: "Energy Management Related Additional Properties" on page 103.

Set BladeCenter Power Save

Use the **Set BladeCenter Power Save** operation to set the power save setting of a BladeCenter.

HTTP method and URI

POST /api/bladecenters/{*bladecenter-id*}/operations/set-power-save

In this request, the URI variable *{bladecenter-id}* is the object ID of the BladeCenter.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`power-saving`	Enum string	Required	The possible settings are: • **"high-performance"** - The power consumption and performance of the BladeCenter are not reduced. This is the default setting. • **"low-power"** - Low power consumption for all components of the BladeCenter enabled for power saving. • **"custom"** - Components may have their own settings changed individually. No component settings are actually changed when this mode is entered.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to control the average energy consumption of a BladeCenter object designated by *{bladecenter-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated BladeCenter is under group control (see “Group capping” on page 140) or the **power-saving** property of the BladeCenter is set to **"not-supported"**or **"not-entitled"**. (See “Energy Management Related Additional Properties” on page 103 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-saving settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described in “HTTP status and reason codes.” After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Save** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in “Response body contents” on page 151.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{bladecenter-id}*) does not designate an existing BladeCenter object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully

Job status codes	Job reason code	Description
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the BladeCenter hardware
	162	Communication error occurred while trying to access the BladeCenter hardware
	163	An error occurred at one or more children

If the job reason code is 163, the **job-results** field provided by the **Query Job Status** operation will contain an object with the following fields:

Field name	Type	Description
`errors`	Object array	A list of error objects, containing detailed error information about errors occurred on children
`at-least-one-operation-succeed`	Boolean	**True** indicates that the operation was successful for at least one child.

Each error object has this structure:

Job status codes	Job reason code	Description
`object-uri`	String URI	The canonical URI path for a specific object where the error occurred
`reason-code`	Integer	Specify the specific error type, possible values are: • 160 - A firmware error occurred while executing the operation • 161 - A hardware error occurred while performing the energy management operation • 162 - Communication error occurred while trying to access the hardware
`message`	String	A non localized message provided for development purposes only. Client applications should not display this message directly to the user.

Set BladeCenter Power Capping

Use the **Set BladeCenter Power Capping** operation to set the power capping settings of a BladeCenter.

HTTP method and URI

`POST /api/bladecenters/{`*`bladecenter-id`*`}/operations/set-power-capping`

In this request, the URI variable *{bladecenter-id}* is the object ID of the BladeCenter.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`power-capping-state`	Enum string	Required	The possible settings are: • **"disabled"** - The power cap of the BladeCenter is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - Capping all components of the BladeCenter available for power capping to limit the peak power consumption of the BladeCenter. • **"custom"** - Individually configure the components of the BladeCenter for power capping. No component settings are actually changed when this mode is entered.
`power-cap-current`	Integer	Optional	Specifies the current cap value for the BladeCenter in watts (W). The current cap value indicates the power budget for the BladeCenter. This field is only required if the **power-capping-state** field is set to **"enabled"**. The **power-cap-current** must be between **power-cap-minimum** and **power-cap-maximum**: **power-cap-minimum** <= value <= **power-cap-maximum**

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to limit the peak power consumption of a BladeCenter object designated by *{bladecenter-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated BladeCenter is under group control (see "Group capping" on page 140) or the **power-capping-state** property of the BladeCenter is set to **"not-supported"** or **"not-entitled"**. (See "Energy Management Related Additional Properties" on page 103 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-capping settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described in "HTTP status and reason codes" on page 155. After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to all blade, BladeCenter, CPC and zCPC objects

- Action/task permission to the **Power Capping** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 154.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
400 (Bad Request)	7	The **power-cap-current** field contains a value that is not in the range **power-cap-minimum** ... **power-cap-maximum**
	5	The **power-cap-current** field is not set, but **power-capping-state** field is set to **"enabled"**.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{bladecenter-id}*) does not designate an existing BladeCenter object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the BladeCenter hardware
	162	Communication error occurred while trying to access the BladeCenter hardware
	163	An error occurred at one or more children

If the job reason code is 163, the **job-results** field provided by the **Query Job Status** operation will contain an object with the following fields:

Field name	Type	Description
`errors`	Object array	A list of error objects, containing detailed error information about errors occurred on children
`at-least-one-operation-succeed`	Boolean	**True** indicates that the operation was successful for at least one child.

Each error object has this structure:

Job status codes	Job reason code	Description
`object-uri`	String URI	The canonical URI path for a specific object where the error occurred
`reason-code`	Integer	Specify the specific error type, possible values are: • 160 - A firmware error occurred while executing the operation • 161 - A hardware error occurred while performing the energy management operation • 162 - Communication error occurred while trying to access the hardware
`message`	String	A non localized message provided for development purposes only. Client applications should not display this message directly to the user.

Energy management for blade object

Data model

The data model for a Blade object includes some properties related to energy management. These properties are described in Chapter 8, "zBX infrastructure elements," on page 69, under the data model for the Blade object: "Energy management related additional properties" on page 114.

Set Blade Power Save

Use the **Set Blade Power Save** operation to set the power save setting of a blade.

HTTP method and URI

`POST /api/blade/{`*`blade-id`*`}/operations/set-power-save`

In this request, the URI variable *{blade-id}* is the object ID of the blade.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`power-saving`	Enum string	Required	The possible settings are: • **"high-performance"** - The power consumption and performance of the blade are not reduced. This is the default setting. • **"low-power"** - Low power consumption for all components of the blade enabled for power saving.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to control the average energy consumption of a blade object designated by *{blade-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated blade is under group control (see "Group capping" on page 140) or the **power-saving** property of the blade is set to **"not-supported"** or **"not-entitled"**. (See "Energy management related additional properties" on page 114 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-saving settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described below. After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Save** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 156.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{blade-id}*) does not designate an existing blade object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully

Job status codes	Job reason code	Description
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the blade hardware
	162	Communication error occurred while trying to access the blade hardware

Set Blade Power Capping

Use the **Set Blade Power Capping** operation to set the power capping settings of a blade.

HTTP method and URI

POST /api/blade/{*blade-id*}/operations/set-power-capping

In this request, the URI variable *{blade-id}* is the object ID of the blade.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
power-capping-state	Enum string	Required	The possible settings are: • **"disabled"** - The power cap of the blade is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - Capping all components of the blade available for power capping to limit the peak power consumption of the blade.
power-cap-current	Integer	Optional	Specifies the current cap value for the blade in watts (W). The current cap value indicates the power budget for the blade. This field is only required if the **power-capping-state** field is set to **"enabled"**. The **power-cap-current** must be between **power-cap-minimum** and **power-cap-maximum**: **power-cap-minimum** <= value <= **power-cap-maximum**

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
job-uri	String	URI of the asynchronous job that may be queried to retrieve status updates for action initiated by this operation.

Description

Use this operation to limit the peak power consumption of a blade object designated by *{bladecenter-id}*, or to remove a power consumption limit for this object. You can closely manage power allocations within the physical limits of your data center.

This operation will always fail if the designated blade is under group control (see “Group capping” on page 140) or the **power-capping-state** property of the blade is set to **"not-supported"** or **"not-entitled"**.

(See “Energy management related additional properties” on page 114 for details on this property.) In addition, this operation is only available if the ensemble is functioning at the Automate management enablement level.

The action to change the power-capping settings occurs asynchronously. If the request is accepted, an asynchronous job is initiated and an HTTP Status code of 202 (Accepted) is returned. The response body includes a URI that may be queried to retrieve the status of the asynchronous job. See the description of the **Query Job Status** operation for information on how to query job status. When the asynchronous job has completed, an asynchronous result message is sent, with Job status and reason codes described below. After completion, the **Query Job Status** operation may be used to retrieve the completion results.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to all blade, BladeCenter, CPC and zCPC objects
- Action/task permission to the **Power Capping** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in “Response body contents” on page 158.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
400 (Bad Request)	7	The **power-cap-current** field contains a value that is not in the range **power-cap-minimum** ... **power-cap-maximum**
	5	The **power-cap-current** field is not set, but **power-capping-state** field is set to **"enabled"**.
403 (Forbidden)	1	The user is not authorized to access the object or perform this task.
	3	The server is not entitled to perform energy management.
404 (Not Found)	1	The object ID in the URI (*{blade-id}*) does not designate an existing blade object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The operation cannot be performed because the object designated by the request URI is not in the correct state.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Job status and reason codes

Job status codes	Job reason code	Description
200 (OK)	N/A	Operation executed successfully

Job status codes	Job reason code	Description
500 (Server Error)	160	A firmware error occurred while executing the operation
	161	A hardware error occurred while performing the operation on the blade hardware
	162	Communication error occurred while trying to access the blade hardware

Chapter 10. Virtualization management

zManager provides facilities for running virtualized computing systems, termed Virtual Servers, on top of system-firmware-managed hosting environments that coordinate the physical system resources that the virtual servers share. These hosting environments upon which the virtual servers run are known as Virtualization Hosts.

The Virtualization Host APIs provide access to the set of virtualization hosts managed by an Ensemble. PowerVM, x Hyp, z/VM and PR/SM™ virtualization hosts are supported. APIs exist to query virtualization hosts, retrieve and update select properties of virtualization hosts, and perform operations on virtualization hosts, such as activate and deactivate.

Within the zEnterprise System, a virtual server can be described as a container for an operating system. It could be a logical partition on an IBM POWER Blade, a virtual machine on an IBM System x Blade, a virtual machine defined under z/VM, or a System z logical partition (LPAR). It is created in cooperation with the virtualization host on the hardware involved. The virtual server container includes the definition of processor resources, network interfaces and storage devices it will access.

Virtualization host operations summary

The following tables provide an overview of the virtualization host operations provided.

Table 32. Virtualization management - virtualization host: operations summary

Operation name	HTTP method and URI path
"List Virtualization Hosts of an Ensemble" on page 173	`GET /api/ensembles/{ensemble-id}/virtualization-hosts`
"List Virtualization Hosts of a CPC" on page 176	`GET /api/cpcs/{cpc-id}/virtualization-hosts`
"Get Virtualization Host Properties" on page 179	`GET /api/virtualization-hosts/{virt-host-id}`
"Update Virtualization Host Properties" on page 183	`POST /api/virtualization-hosts/{virt-host-id}`
"List Virtual Switches" on page 184	`GET /api/virtualization-hosts/{virt-host-id}/virtual-switches`
"Get Virtual Switch Properties" on page 186	`GET /api/virtualization-host/{virt-host-id}/virtual-switches/{virtual-switch-id}`
"Create IEDN Virtual Switch" on page 189	`POST /api/virtualization-hosts/{virt-host-id}/virtual-switches/operations/add-iedn`
"Create QDIO Virtual Switch" on page 192	`POST /api/virtualization-hosts/{virt-host-id}/virtual-switches/operations/add-qdio`
"Get Switch Controllers" on page 195	`GET /api/virtualization-hosts/{virt-host-id}/operations/get-switch-controllers`
"Update Virtual Switch" on page 197	`POST /api/virtualization-hosts/{virt-host-id}/virtual-switches/{virtual-switch-id}`
"Delete Virtual Switch" on page 201	`DELETE /api/virtualization-hosts/{virt-host-id}/virtual-switches/{virtual-switch-id}`

Table 33. Virtualization management- virtualization host: URI variables

Variable	Description
{ensemble-id}	Object ID of an ensemble object
{cpc-id}	Object ID of a CPC object
{virt-host-id}	Object ID of a virtualization host object
{virtual-switch-id}	Element ID of a virtual switch

Virtual server operations summary

The following tables provide an overview of the virtual server operations provided.

Table 34. Virtualization management - virtual server: operations summary

Operation name	HTTP method and URI path
"List Virtual Servers of an Ensemble" on page 230	`GET /api/ensembles/{ensemble-id}/virtual-servers`
"List Virtual Servers of a CPC" on page 232	`GET /api/cpcs/{cpc-id}/virtual-servers`
"List Virtual Servers of a Virtualization Host" on page 235	`GET /api/virtualization-hosts/{virt-host-id}/virtual-servers`
"Create Virtual Server" on page 237	`POST /api/virtualization-hosts/{virt-host-id}/virtual-servers`
"Delete Virtual Server" on page 242	`DELETE /api/virtual-servers/{virtual-server-id}`
"Get Virtual Server Properties" on page 243	`GET /api/virtual-servers/{virtual-server-id}`
"Update Virtual Server Properties" on page 251	`POST /api/virtual-servers/{virtual-server-id}`
"Create Network Adapter" on page 256	`POST /api/virtual-servers/{virtual-server-id}/network-adapters`
"Update Network Adapter" on page 259	`POST /api/virtual-servers/{virtual-server-id}/network-adapters/{network-adapter-id}`
"Delete Network Adapter" on page 262	`DELETE /api/virtual-servers/{virtual-server-id}/network-adapters/{network-adapter-id}`
"Reorder Network Adapter" on page 263	`POST /api/virtual-servers/{virtual-server-id}/operations/reorder-network-adapters`
"Create Virtual Disk" on page 265	`POST /api/virtual-servers/{virtual-server-id}/virtual-disks`
"Delete Virtual Disk" on page 268	`DELETE /api/virtual-servers/{virtual-server-id}/virtual-disks/{virtual-disk-id}`
"Get Virtual Disk Properties" on page 270	`GET /api/virtual-servers/{virtual-server-id}/virtual-disks/{virtual-disk-id}`
"Update Virtual Disk Properties" on page 272	`POST /api/virtual-servers/{virtual-server-id}/virtual-disks/{virtual-disk-id}`
"Reorder Virtual Disks" on page 274	`POST /api/virtual-servers/{virtual-server-id}/operations/reorder-virtual-disks`
"Activate Virtual Server" on page 277	`POST /api/virtual-servers/{virtual-server-id}/operations/activate`

Table 34. Virtualization management - virtual server: operations summary (continued)

Operation name	HTTP method and URI path
"Deactivate Virtual Server" on page 278	`POST /api/virtual-servers/{virtual-server-id}/operations/deactivate`
"Mount Virtual Media" on page 280	`POST /api/virtual-servers/{virtual-server-id}/operations/mount-virtual-media`
"Mount Virtual Media Image" on page 283	`POST /api/virtual-servers/{virtual-server-id}/operations/mount-virtual-media-image`
"Unmount Virtual Media" on page 285	`POST /api/virtual-servers/{virtual-server-id}/operations/unmount-virtual-media`
"Migrate Virtual Server" on page 286	`POST /api/virtual-servers/{virtual-server-id}/operations/migrate`
"Initiate Virtual Server Dump" on page 289	`POST /api/virtual-servers/{virtual-server-id}/operations/initiate-dump`

Table 35. Virtualization management - virtual server: URI variables

Variable	Description
{ensemble-id}	Object ID of an ensemble object
{cpc-id}	Object ID of a CPC object
{virt-host-id}	Object ID of a virtualization host object
{virtual-server-id}	Object ID of a virtual server object
{network-adapter-id}	Element ID of an network adapter object
{virtual-disk-id}	Element ID of a virtual disk object

Virtualization host object

A virtualization host object represents a single zEnterprise virtualization host.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, with the following class-specific specialization:

Table 36. Virtualization host object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path for a virtualization host object is of the form `/api/virtualization-hosts/{object-id}`.
name	(ro)	String (1-64)	The display name of the virtualization host.[1] Not writeable. The name is derived from the name of the underlying hosting environment (e.g. blade).
class	—	String (19)	The class will always be **"virtualization-host"**.
parent	—	String/ URI	The canonical URI path of the node that manages the virtualization host.

Table 36. Virtualization host object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
status	(sc)	String Enum	The current operational status of the virtualization host. Valid values common to all types: • **"operating"** - indicates virtualization host is running normally • **"not-operating"** - indicates the virtualization host is deactivated, in the process of loading, or has a error • **"status-check"** - indicates the support element is not communicating with the CPC • **"not-communicating"** - indicates the HMC is not communicating with the support element. Valid values for type **"power-vm"**, **"x-hyp"**: • **"definition-error"** - indicates that the specified zBX Blade does not match the characteristics of the installed zBX Blade • **"no-power"** - indicates the zBX Blade powered off • **"stopped"** - indicates the Support Element with the zBX Blade is stopped. Valid values for type **"zvm"**: • **"exceptions"** - indicates at least one CP is operating, but at least one CP is not operating. • **"not-activated"** - indicates the image has not been activated. Valid values for type **"prsm"**: • **"degraded"** - indicates the CPC is operating but some hardware is not available. • **"exceptions"** - indicates that at least one Central Processor (CP) is operating, but at least one CP is not operating. • **"no-power"** - indicates that the CPC is powered off. • **"service"** - indicates that CPs are in service status. • **"service-required"** - indicates that the CPC is still operating but is using the last redundant part of a particular type.
`additional status`	—		This property is not provided.

Note: [1]The location of a blade can be moved from slot to slot within a zBX. When a blade is moved to a different slot, the original URI of this blade and virtualization host object is retained. Since the name of the blade and virtualization host is based on the slot location of the blade, the **name** property can change for a given URI when the blade is moved within the zBX. The relocation of a blade generates inventory change notifications to report the removal of the blade and the corresponding virtualization host object, then inventory change notifications to report the addition of the blade and the virtualization host. Upon addition of the blade and virtualization host, expect the value of the **name** property to differ.

Class specific additional properties

In addition to the properties defined in included schemas, this object includes the following additional type-specific properties:

Table 37. Virtualization host object: class specific additional properties

Name	Qualifier	Type	Description	Supported "type" values
type	—	String Enum	Type of the virtual server. Values: • **"power-vm"**- a virtualization host running on a POWER blade • **"x-hyp"** - The canonical URI path for the System x blade that hosts the virtualization host. • **"zvm"** - an IBM z/VM operating system instance that is participating as an ensemble-managed virtualization host • **"prsm"** - the virtualization host representation of a CPC	All
hosting-environment	—	String/URI	The hosting environment (cpc, image, or blade) of the virtualization host. Value based on type: • **"power-vm"** - The canonical URI path for the System z POWER blade that hosts the virtualization host. • **"x-hyp"** - The canonical URI path for the System x Blade that hosts the virtualization host. • **"zvm"** - The canonical URI path for the PR/SM virtual server that is hosting the z/VM virtualization host. • **"prsm"** - The canonical URI path for the CPC that is the PR/SM virtualization host.	All
virtual-server-shutdown-timeout	(w)(pc)	Integer	Amount of time, in seconds, to allow a virtual server to cleanly shutdown. After the elapsed time has passed, the virtual server will be forcefully stopped. The value may be -1 to indicate to wait "forever" or any integer value between 0 and (2^31 – 1) to specify an exact wait time in seconds.	All
auto-start-virtual-servers	(w)(pc)	Boolean	Automatically start any virtual servers configured to start when the virtualization host is started.	power-vm, x-hyp
total-memory	(pc)	Integer	The total amount of memory the hypervisor has (in MB).	All
memory-increment-in-megabytes	—	Integer	The minimum "increment", in megabytes, that is required when setting the memory size for a virtual server.	power-vm, x-hyp, zvm
minimum-memory-size-for-virtual-server	—	Integer	The minimum memory size, in megabytes, that can be configured for a virtual server.	power-vm, x-hyp, zvm
maximum-memory-size-for-virtual-servers	—	Integer	The maximum memory size, in megabytes, that can be configured for a virtual server.	power-vm, x-hyp, zvm
maximum-allowed-dedicated-processors	—	Integer	The maximum number of dedicated processors that can be configured for a virtual server.	power-vm, zvm

Table 37. Virtualization host object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
maximum-allowed-virtual-processors	—	Integer	The maximum number of virtual processors that can be configured for a virtual server.	power-vm, x-hyp, zvm
maximum-allowed-processing-units	—	Float	The maximum number of processing units that can be configured for a virtual server. Processing units are a unit of measure for shared processing power across one or more virtual processors. One shared processing unit on one virtual processor accomplishes approximately the same work as one dedicated processor.	power-vm
maximum-allowed-ide-devices	—	Integer	The maximum number of IDE devices that can be configured for a virtual server	x-hyp
mixed-mode-boot-restriction	—	Boolean	Indicates if a virtual disk boot restriction exists for the virtualization host's virtual servers that would prevent booting from **"virtio"** virtual disks when both **"ide"** and **"virtio"** virtual disks are defined. In such a case, the virtual server will always boot from an **"ide"** virtual disk.	x-hyp
inband-monitoring-supported	—	Boolean	If true, in-band monitoring is supported for the Virtualization Host, allowing it to gather performance metrics on its virtual servers.	power-vm, x-hyp
supported-keyboard-languages	—	Array of Strings	List of keyboard languages supported by the hypervisor for graphical console connections. The value is **null** if no key map is supported or necessary or if graphical console is not supported. See the virtual server object's **keyboard-language** property for the description of an array element value.	x-hyp
is-bridge-capable	—	Boolean	If the virtualization host is capable of having virtual switches with bridge ports, the value is true. For true to be returned the following requirements must be met; otherwise, false is returned: • The **os-level** property of the PR/SM virtual server running the z/VM virtualization host is "621" or greater • The **se-version** property of the CPC object is 2.11.1 or higher.	zvm

Table 37. Virtualization host object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
feature-list	—	Array of String Enums	Optional features or behaviors supported by this virtualization host. The presence of one of the Enum values described below indicates that this virtualization host provides the described feature. If the virtualization host has no such optional features, an empty array is provided. Backport related **feature-list** values for an **"x-hyp"** virtualization host: • **"boot-sequence-mixed-disk-restriction"** - Indicates that a virtual disk boot restriction exists for the virtualization host's virtual servers that would prevent booting from **"virtio"** virtual disks when both **"ide"** and **"virtio"** virtual disks are defined. In such a case, the virtual server will always boot from an **"ide"** virtual disk. • **"boot-sequence-network-priority-restriction"** - Indicates that a virtual network adapter boot restriction exists for the virtualization host's virtual servers that would force **"network-adapter"** to be the first entry in the **boot-sequence** property. • **"boot-sequence-virtual-disk-enforcement"** - Indicates that a virtualization host's virtual server's **boot-sequence** property must contain an entry for **"virtual-disk"** if at least one virtual disk has been configured. • **"boot-sequence-network-adapter-enforcement"** - Indicates that a virtualization host's virtual server's **boot-sequence** property must contain an entry for **"network-adapter"** if at least one network adapter has been configured. • **"boot-sequence-virtual-media-enforcement"** - Indicates that a virtualization host's virtual server's **boot-sequence** property must contain an entry for **"virtual-media"**.	All

Virtual switch objects

A virtual switch is a special type of guest LAN that provides external LAN connectivity through an OSA-Express device without the need for a routing virtual machine.

Note: Some properties are only valid when mutable prerequisite properties have specific values. When such properties are not valid, their value is **null**. For instance a **iedn-virtual-switch** object's router property value is **null** when the **layer-mode** value is **"eth"**.

iedn-virtual-switch object: The iedn-virtual-switch object is a virtual-switch object that defines the IEDN virtual switch of a virtualization-host object of type **"zvm"**.

Table 38. iedn-virtual-switch object properties

Name	Qualifier	Type	Description
element-id	—	String	Unique ID for the virtual switch within the scope of the containing virtualization host. The element-id is actually the name of the virtual switch. Once the virtual switch is created, the name cannot be changed.
element-uri	—	String/URI	The canonical URI path for the virtual switch is of the form `/api/virtualization-hosts/{virt-host-id}/virtual-switches/{element-id}`, where *{virt-host-id}* is the `object-id` of the virtualization host.
parent	—	String/URI	The URI path of the z/VM Virtualization Host that hosts this virtual switch.
class	—	String (14)	Always **"virtual-switch"**
type	—	String Enum	The virtual switch type. Always **"iedn"**
name	—	String (1-8)	The unique name identifying the virtual switch. 1-8 alphanumeric uppercase characters or any of the following characters: “@#$”
switch-status	—	String	Virtual switch status, read-only.
layer-mode	—	String Enum	Indicates the transport for the virtual switch is Ethernet or IP. Values: • **"eth"** : This type is Data Link (Layer 2) based, where the Ethernet frame is used as the point of reference for source and destination Media Access Control (MAC) addresses in transporting Ethernet frames on the LAN. • **"ip"**: This type is Network (Layer 3) based, where the IP package is used as the point of reference for source and destination IP addresses in transporting IP packets on the LAN.
router	—	String Enum	The router type. Prerequisites: layer-mode is **"ip"**. Values: • **"none"** - Indicates that the OSA-Express device identified will not act as a router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram will be discarded. • **"primary"** - Indicates that the OSA-Express device identified will act as a primary router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram will be passed to the virtual switch. The only time to set PRIMARY for a virtual switch is if you have a guest attached to the virtual switch that is providing a routing function for systems attached to another network.
queue-size	—	Integer	Queue storage in megabytes: decimal number between 1 and 8
ip-timeout	—	Integer	IP timeout in seconds: decimal number between 1 and 240
is-connect-uplinks	—	Boolean	If the switch is connected to uplinks, the value is true.
real-uplinks	—	Array of real-uplink objects	A list of zero to three real-uplink objects.
is-use-any-available-controller	—	Boolean	True if the switch uses any available controller.

Table 38. iedn-virtual-switch object properties (continued)

Name	Qualifier	Type	Description
controllers	—	Array of strings	A list of names of switch controllers. Prerequisites: **is-use-any-available-controller** is false.
bridge-value-type	(w)	String Enum	Whether the bridge port is a primary or secondary port. There can only be 1 primary and up to 4 secondary bridge ports. Values: • **"primary"** - Bridge port is the primary port • **"secondary"** - Bridge port is a secondary port. Prerequisite: This field is applied to a vSwitch that has a valid **bridge-device-number**. If not specified when the bridge port is defined, then the default is **"secondary"**. This field can be modified when the **bridge-connection-status** is **"disconnected"**.
bridge-device-number	(w)	String (1-4)	The bridge device number. The following values may be specified: • The bridge real device number, 1~4 hex digits. Prerequisites: The **is-bridge-capable** property of the virtualization host is true and the **layer-mode** property of the virtual switch is **"eth"**. This property is required to define a device for a bridge port and must be a nonblank, valid device number.
bridge-connection-status	(w)	String Enum	Whether the bridge port is connected or in standby state. Prerequisites: The **is-bridge-capable** property of the virtualization host is true and the virtual switch has a valid **bridge-device-number**. Values: • **"connected"** - Bridge port is connected • **"disconnected"** - Bridge port is disconnected. • **"standby"** - Bridge port is in standby state
mtu-size-enforcement	(w)	String Enum	How the virtual switch MTU size is enforced. Prerequisites: The **is-bridge-capable** property of the virtualization host is true. Values: • **"external"** - The MTU size will be set to the size used by the OSA adapter. This is the default value. • **"off"** - MTU enforcement is disabled • **"user-defined"** - The MTU size is specified in the **mtu-size** field
mtu-size	(w)	Integer	The MTU specification represents the acceptable MTU size, in bytes, enforced by the virtual switch. MTU size is a decimal number between 512-65535. Prerequisite: This field is applied to a virtual switch where the **is-bridge-capable** property of the virtualization host is true and the **mtu-size-enforcement** property of the virtual switch is **"user-defined"**.

real-uplink object: This is the embedded object definition for a virtual switch. For IEDN virtual switch, the **switch-uri** property is required. For QDIO virtual switch, the **switch-uri** property is ignored.

Table 39. real-uplink object properties

Name	Type	Description
switch-uri	String	The **element-uri** of the network-adapter-prsm object in use by the z/VM virtual switch to provide a connection between a virtual network and a physical network. Prerequisites: **type** in the parent switch is **"iedn"**.
device-number	String (1-8)	The uplink real device number, 1-8 hex digits. The value could be either **"none"** or a string with the regular expression pattern [0-9a-fA-F]{1,4}(.P[0-9][0-9])?. If the optional (.P[0-9][0-9]) part is not specified, a default string ".P00" will be used.

qdio-virtual-switch object

The qdio-virtual-switch object is a virtual-switch object that defines the QDIO virtual switch of a virtualization-host object of type **"zvm"**.

Table 40. qdio-virtual-switch object properties

Name	Qualifier	Type	Description
element-id	—	String	Unique ID for the virtual switch within the scope of the containing virtualization host. The element-id is actually the name of the virtual switch. Once the virtual switch is created, the name cannot be changed.
element-uri	—	String/URI	The canonical URI path for the virtual switch is of the form `/api/virtualization-hosts/{`*`virt-host-id`*`}/virtual-switches/{`*`element-id`*`}`, where *{virt-host-id}* is the **object-id** of the virtualization host.
parent	—	String/URI	The URI path of the z/VM Virtualization Host that hosts this virtual switch.
class	—	String (14)	Always **"virtual-switch"**
type	—	String Enum	Virtual switch type. Always **"qdio"**
name	—	String (1-8)	The unique name identifying the virtual switch. 1-8 alphanumeric uppercase characters, or any of the following characters: “@#$”
switch-status	—	String	Virtual switch status, read-only.
is-vlan-aware	—	Boolean	True if not VLAN unaware. VLAN unaware is a classification for a networking device that indicates it does not support the IEEE 802.1Q VLAN specification for VLAN membership and VLAN frame formats. These devices ignore the additional fields within the Ethernet frame that carry VLAN specific semantics.
vlan-id	—	String (1-4)	Vlan ID : 1-4 decimal digits, maximum 4094. Prerequisites: **is-vlan-aware** is true.
vlan-object-uri	—	String/URI	The canonical URI path for the associated virtual network. Prerequisites: **is-vlan-aware** is true.

Table 40. qdio-virtual-switch object properties (continued)

Name	Qualifier	Type	Description
vlan-port-type	—	String Enum	Indicates the port type of the simulated NIC. This setting is applied to the switch, not the single port on the switch. Prerequisites: **is-vlan-aware** is true. Values: • **"trunk"**: When the switch port is configured in trunk mode, it will allow the flow of traffic from multiple virtual networks (i.e. VLANS). The port must be configured with those virtual networks. • **"access"**: When the switch port is configured in access mode, it will support a single virtual network. Traffic from the virtual server's network adapter will be tagged with the virtual network configured for this port, and traffic destined to the virtual server on this port will be verified that it is tagged with the configured virtual network.
vlan-native-id	—	String	Native vlan ID : 1-4 decimal digits, maximum 4094. Prerequisites: **is-vlan-aware** is true. A native VLAN ID, (usually VLAN ID 0001) is deployed internally by the virtual switch to associate or flow untagged frames through the switching fabric. Only those guests that are configured for the native VLAN ID will receive or send untagged frames.
is-gvrp-enabled	—	Boolean	True if GVRP is enabled. Prerequisites: **is-vlan-aware** is true. Generic Attribute Registration Protocol (GARP) VLAN Registration Protocol (GVRP) is an application defined in the IEEE802.1Q standard that allows for the control of VLANs. It runs only on 802.1Q trunk links. It prunes trunk links so that only active VLANs will be sent across trunk connections.
layer-mode	—	String Enum	Indicates the transport for the virtual switch is Ethernet or IP. Values: • **"eth"** : This type is Data Link (Layer 2) based, where the Ethernet frame is used as the point of reference for source and destination Media Access Control (MAC) addresses in transporting Ethernet frames on the LAN. • **"ip"**: This type is Network (Layer 3) based, where the IP package is used as the point of reference for source and destination IP addresses in transporting IP packets on the LAN.

Table 40. qdio-virtual-switch object properties (continued)

Name	Qualifier	Type	Description
router	—	String Enum	The router type. Prerequisites: layer-mode is **"ip"**. Values: • **"none"** - Indicates that the OSA-Express device identified will not act as a router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram will be discarded. • **"primary"** - Indicates that the OSA-Express device identified will act as a primary router to the virtual switch. If a datagram is received at this device for an unknown IP address, the datagram will be passed to the virtual switch. The only time to set PRIMARY for a virtual switch is if you have a guest attached to the virtual switch that is providing a routing function for systems attached to another network.
queue-size	—	Integer	Indicates the upper limit of the amount of fixed storage CP and Queued Direct I/O Hardware Facility will use for buffers for each OSA-Express data device. Queue storage in megabytes: decimal number between 1 and 8.
ip-timeout	—	Integer	IP timeout in seconds: decimal number between 1 and 240.
is-connect-uplinks		Boolean	True if connected to uplinks.
real-uplinks	—	Array of real-uplink objects	A list of zero to three real-uplink objects.
is-use-any-available-controller	—	Boolean	True if the switch uses any available controller.
controllers	—	Array of strings	A list of names of switch controllers. Prerequisites: **is-use-any-available-controller** is false.
bridge-value-type	(w)	String Enum	Whether the bridge port is a primary or secondary port. There can only be 1 primary and up to 4 secondary bridge ports. Values: • **"primary"** - Bridge port is the primary port • **"secondary"** - Bridge port is a secondary port. Prerequisite: This field is applied to a vSwitch that has a valid **bridge-device-number**. If not specified when the bridge port is defined, then the default is **"secondary"**. This field can be modified when the **bridge-connection-status** is **"disconnected"**.
bridge-device-number	(w)	String (1-4)	The bridge device number. The following values may be specified: • The bridge real device number, 1~4 hex digits. Prerequisites: The **is-bridge-capable** property of the virtualization host is true and the **layer-mode** property of the virtual switch is **"eth"**. This property is required to define a device for a bridge port and must be a nonblank, valid device number.

Table 40. qdio-virtual-switch object properties (continued)

Name	Qualifier	Type	Description
bridge-connection-status	(w)	String Enum	Whether the bridge port is connected, disconnected or in standby state. Prerequisites: The **is-bridge-capable** property of the virtualization host is true and the virtual switch has a valid **bridge-device-number**. Values: • **"connected"** - Bridge port is connected (default) • **"disconnected"** - Bridge port is disconnected • **"standby"** - Bridge port is in standby state.
mtu-size-enforcement	(w)	String Enum	How the virtual switch MTU size is enforced. Prerequisites: The **is-bridge-capable** property of the virtualization host is true. Values: • **"external"** - The MTU size will be set to the size used by the OSA adapter. This is the default value. • **"off"** - MTU enforcement is disabled • **"user-defined"** - The MTU size is specified in the **mtu-size** field
mtu-size	(w)	Integer	The MTU specification represents the acceptable MTU size, in bytes, enforced by the virtual switch. MTU size is a decimal number between 512-65535. Prerequisite: This field is applied to a virtual switch where the **is-bridge-capable** property of the virtualization host is true and the **mtu-size-enforcement** of the virtual switch is **"user-defined"**.

Operations

If a virtualization host operation accesses a z/VM virtualization host and encounters an error while communicating with the virtualization host via SMAPI, the response body is a SMAPI Error Response Body.

List Virtualization Hosts of an Ensemble

The **List Virtualization Hosts of an Ensemble** operation lists the Virtualization Hosts managed by the ensemble with the given identifier.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/virtualization-hosts

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.
type	String Enum	Optional	Filter string to limit returned objects to those that have a matching **type** property. Value must be a valid Virtualization Host **type** property value.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
virtualization-hosts	Array of objects	Array of virtualization-host-info objects, described in the next table. Returned array may be empty.

Each nested virtualization-host-info object contains the following fields:

Field name	Type	Description
object-uri	String/ URI	The canonical URI path of the Virtualization Host
name	String	Name of the Virtualization Host
type	String Enum	Type of Virtualization Host
status	String Enum	Current status of the Virtualization Host

Description

This operation lists the Virtualization Hosts that are managed by the identified ensemble.

If the **name** query parameter is specified, the returned list is limited to those Virtualization Hosts that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done

If the **type** query parameter is specified, the parameter is validated to ensure it is a valid Virtualization Host **type** property value. If the value is not valid, a 400 (Bad Request) is returned. If the value is valid, the returned list is limited to those Virtualization Hosts that have a **type** property matching the specified value. If the **type** parameter is omitted, this filtering is not done.

A Virtualization Host is included in the list only if the API user has object-access permission to its hosting-environment. If an HMC is a manager of a Virtualization Host but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the ensemble does not manage any Virtualization Hosts or if no Virtualization Hosts are to be included in the results due to filtering, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the specified ensemble
- Object access permission to hosting-environment of the Virtualization Hosts managed by the specified ensemble.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is provided and the response body is as described in “Response body contents.”

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	80	A **type** query parameter defines an invalid value.
404 (Not Found)	1	The request URI does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/virtualization-hosts HTTP/1.1
x-api-session: 4oup923zgs27vmd1wpzvu47ikgzhxa8bwimjofpq6d3eq3j13q
```

Figure 71. List Virtualization Hosts of an Ensemble: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 824
{
   "virtualization-hosts": [
      {
         "name": "APIVM1",
         "object-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4",
         "status": "operating",
         "type": "zvm"
      },
      {
         "name": "B.2.02",
         "object-uri": "/api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de",
         "status": "operating",
         "type": "power-vm"
      },
      {
         "name": "B.2.03",
         "object-uri": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d",
         "status": "operating",
         "type": "x-hyp"
      },
      {
         "name": "R32",
         "object-uri": "/api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de",
         "status": "operating",
         "type": "prsm"
      }
   ]
}
```

Figure 72. List Virtualization Hosts of an Ensemble: Response

List Virtualization Hosts of a CPC

The **List Virtualization Hosts of a CPC** operation lists the Virtualization Hosts managed by the CPC with the given identifier.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/virtualization-hosts

In this request, the URI variable *{cpc-id}* is Object ID of the CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.
type	String Enum	Optional	Filter string to limit returned objects to those that have a matching **type** property. Value must be a valid Virtualization Host **type** property value.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`virtualization-hosts`	Array of objects	Array of virtualization-host-info objects, described in the next table. Returned array may be empty.

Each nested virtualization-host-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	The canonical URI path of the Virtualization Host
`name`	String	Name of the Virtualization Host object
`type`	String Enum	Type of Virtualization Host. (**"power-vm"**, **"x-hyp"**, **"zvm"**, or **"prsm"**)
`status`	String Enum	Current status of the Virtualization Host

Description

This operation lists the Virtualization Hosts that are managed by the specified CPC. The object URI, object ID, display name, and display description are provided for each.

If the **name** query parameter is specified, the returned list is limited to those Virtualization Hosts that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done

If the **type** query parameter is specified, the parameter is validated to ensure it is a valid Virtualization Host **type** property value. If the value is not valid, a 400 (Bad Request) is returned. If the value is valid, the returned list is limited to those Virtualization Hosts that have a **type** property matching the specified value. If the **type** parameter is omitted, this filtering is not done.

If both **name** and **type** query parameters are specified, a Virtualization Host is included in the list only if it passes both the **name** and **type** filtering criteria.

A Virtualization Host is included in the list only if the API user has object-access permission to its hosting-environment. If an HMC is a manager of a Virtualization Host but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

The list that is returned is never empty because a CPC always has a PR/SM Virtualization Host, and may have additional ones as well. If no Virtualization Hosts are to be included in the results due to filtering, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the specified CPC
- Object access permission to hosting-environment of the Virtualization Hosts managed by the specified CPC.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is provided and the response body is as described in "Response body contents" on page 177.

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	80	A **type** query parameter defines an invalid value.
404 (Not Found)	1	The request URI does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/virtualization-hosts HTTP/1.1
x-api-session: 4oup923zgs27vmd1wpzvu47ikgzhxa8bwimjofpq6d3eq3j13q
```

Figure 73. List Virtualization Hosts of a CPC: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 824
{
   "virtualization-hosts": [
      {
         "name": "APIVM1",
         "object-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4",
         "status": "operating",
         "type": "zvm"
      },
      {
         "name": "B.2.02",
         "object-uri": "/api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de",
         "status": "operating",
         "type": "power-vm"
      },
      {
         "name": "B.2.03",
         "object-uri": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d",
         "status": "operating",
         "type": "x-hyp"
      },
      {
         "name": "R32",
         "object-uri": "/api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de",
         "status": "operating",
         "type": "prsm"
      }
   ]
}
```

Figure 74. List Virtualization Hosts of a CPC: Response

Get Virtualization Host Properties

The **Get Virtualization Host Properties** operation retrieves the properties of a single Virtualization Host object that is designated by its object-id.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}

In this request, the URI variable *{virt-host-id}* is Object ID of the Virtualization Host object for which properties are to be obtained.

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the Virtualization Host object as defined in the "Data model" on page 163. Field names and data types in the JSON object are the same as the property and data types defined in the data model.

Description

Retrieve the current values for properties supported by the specified Virtualization Host.

If the object-id *{virt-host-id}* does not identify a Virtualization Host object for which the API user has object-access permission to its hosting-environment, a 404 status code is returned.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is provided and the response body is as described in "Response body contents" on page 179.

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The request URI does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de HTTP/1.1
x-api-session: 4oup923zgs27vmd1wpzvu47ikgzhxa8bwimjofpq6d3eq3j13q
```

Figure 75. Get Virtualization Host Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 640
{
   "acceptable-status": [
      "operating"
   ],
   "class": "virtualization-host",
   "description": "Initial description",
   "has-unacceptable-status": false,
   "hosting-environment": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "is-locked": false,
   "name": "R32",
   "object-id": "bab76208-2990-11e0-8d5b-001f163803de",
   "object-uri": "/api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
    be79e49dd340",
   "status": "operating",
   "type": "prsm",
   "virtual-server-shutdown-timeout": 200
}
```

Figure 76. Get Virtualization Host Properties: Response for virtualization host of type "prsm"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 988
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start-virtual-servers": true,
   "class": "virtualization-host",
   "description": "single update ABC ",
   "has-unacceptable-status": true,
   "hosting-environment": "/api/blades/938706AC3FF111D78B5600215EC0330E",
   "is-locked": false,
   "maximum-allowed-dedicated-processors": 57,
   "maximum-allowed-processing-units": 57.600000000000001,
   "maximum-allowed-virtual-processors": 64,
   "maximum-memory-size-for-virtual-server": 27648,
   "memory-increment-in-megabytes": 256,
   "minimum-memory-size-for-virtual-server": 256,
   "name": "B.2.02",
   "object-id": "ba97ff30-2990-11e0-8d5b-001f163803de",
   "object-uri": "/api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
    be79e49dd340",
   "status": "operating",
   "type": "power-vm",
   "virtual-server-shutdown-timeout": 2147483647
}
```

Figure 77. Get Virtualization Host Properties: Response for virtualization host of type "power-vm"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 930
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start-virtual-servers": true,
   "class": "virtualization-host",
   "description": "",
   "has-unacceptable-status": true,
   "hosting-environment": "/api/blades/B8210BC02D1E11E0AE81E41F13FE1430",
   "is-locked": false,
   "maximum-allowed-ide-devices": 3,
   "maximum-allowed-virtual-processors": 16.0,
   "maximum-memory-size-for-virtual-server": 125829,
   "memory-increment-in-megabytes": 1,
   "minimum-memory-size-for-virtual-server": 1,
   "mixed-mode-boot-restriction": true,
   "name": "B.2.03",
   "object-id": "931b25d6-82e1-11e0-b9e4-f0def10bff8d",
   "object-uri": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
    be79e49dd340",
   "status": "operating",
   "type": "x-hyp",
   "virtual-server-shutdown-timeout": 302
}
```

Figure 78. Get Virtualization Host Properties: Response for virtualization host of type "x-hyp"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:37 GMT
content-type: application/json;charset=UTF-8
content-length: 893
{
   "acceptable-status": [
      "operating"
   ],
   "class": "virtualization-host",
   "description": "An initial description",
   "has-unacceptable-status": false,
   "hosting-environment": "/api/virtual-servers/4401b16c-ac9a-11e0-ade4-001f163805d8",
   "is-locked": false,
   "maximum-allowed-dedicated-processors": 64,
   "maximum-allowed-virtual-processors": 64,
   "maximum-memory-size-for-virtual-server": 1048576,
   "memory-increment-in-megabytes": 1,
   "minimum-memory-size-for-virtual-server": 64,
   "name": "APIVM1",
   "object-id": "342d80e0-65ff-11e0-acfd-f0def10c03f4",
   "object-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
    be79e49dd340",
   "status": "operating",
   "type": "zvm",
   "virtual-server-shutdown-timeout": 212
}
```

Figure 79. Get Virtualization Host Properties: Response for virtualization host of type "zvm"

Update Virtualization Host Properties

The **Update Virtualization Host Properties** operation updates one or more of the writeable properties of a Virtualization Host.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host object for which properties are to be updated.

Request body contents

The request body contains a JSON object that provides the new values of the writeable properties of the Virtualization Host object as defined in the Data Model section above. Field names and data types in the JSON object are the same as the property or relationship names and data types defined in the data model.

Writeable properties are only valid if they are supported for a Virtualization Host whose **type** property matches the given **type** property value. For example, auto-start-virtual-servers is only a writeable property for type **"power-vm"** and **"x-hyp"** Virtualization Hosts.

Description

This operation updates writeable properties of the Virtualization Host object specified by the request URI.

The request body contains an object with one or more fields with field names that correspond to the names of properties for this object. On successful execution, the value of each corresponding property of the object is updated with the value provided by the input field, and status code 204 (No Content) is returned without supplying any response body. The request body does not need to specify a value for all writeable properties, but rather can and should contain fields for the properties to be updated. Object properties for which no input value is provided remain unchanged by this operation.

If the update changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

The URI path must designate an existing Virtualization Host object and the API user must have object-access permission to its hosting-environment. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have task permission to the **Hypervisor Details** task as well, otherwise status code 403 (Forbidden) is returned.

The request body is validated against the data model for this object type to ensure that it contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Action/task permission to the **Hypervisor Details** task
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The request URI does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

List Virtual Switches

The **List Virtual Switches** operation lists the virtual switches managed by the z/VM Virtualization Host with the given identifier.

HTTP method and URI

GET /api/virtualization-host/{*virt-host-id*}/virtual-switches

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
virtual-switches	Array of objects	Array of virtual-switch-info objects, described in the next table. Returned array may be empty.

Each nested virtual-switch-info object contains the following fields:

Field name	Type	Description
element-uri	String/URI	The **element-uri** property of virtual-switch element.
type	String Enum	The **type** property of virtual-switch element.
name	String	The **name** property of virtual-switch element.

Description

This operation lists the virtual switches that are managed by the identified Virtualization Host.

If the Virtualization Host does not have any virtual network switches, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Action/task role permission to the **Manage Virtual Switches** task
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/virtual-switches HTTP/1.1
x-api-session: 4oup923zgs27vmd1wpzvu47ikgzhxa8bwimjofpq6d3eq3j13q
```

Figure 80. List Virtual Switches: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 22 Jul 2011 15:02:46 GMT
content-type: application/json;charset=UTF-8
content-length: 411
{
   "virtual-switches": [
      {
         "element-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/virtual-switches/
       S2777",
         "name": "S2777",
         "type": "iedn"
      },
      {
         "element-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/virtual-switches/
       4179SW3",
         "name": "4179SW3",
         "type": "qdio"
      }
   ]
}
```

Figure 81. List Virtual Switches: Response

Get Virtual Switch Properties

The **Get Virtual Switch Properties** operation retrieves the properties of a single virtual switch that is designated by the request URI.

HTTP method and URI

GET /api/virtualization-host/{*virt-host-id*}/virtual-switches/{*virtual-switch-id*}

URI variables

Variable	Type	Description
virt-host-id	String	Object ID of the Virtualization Host
virtual-switch-id	String	Element ID of the Virtual Switch

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the virtual switch object as defined in the "Data model" on page 163. Field names and data types in the JSON object are the same as the property or relationship names and data types defined in the data model.

Description

This operation returns a JSON object that provides the current values of the properties for the virtual switch element object as defined in the"Data model" on page 163.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*
- Task role permission to the **Manage Virtual Switches** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/57ab94c8-03e6-11e1-baf3-001f163805d8/virtual-switches/SSSS HTTP/1.1
x-api-session: 3ch9r8g2lavxw9st52brubgk4bpsqik3jcqbg8hdwpkg5flwpx
```

Figure 82. Get Virtual Switch Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 06:01:18 GMT
content-type: application/json;charset=UTF-8
content-length: 575
{
   "bridge-connection-status": null,
   "bridge-device-number": null,
   "bridge-value-type": null,
   "class": "virtual-switch",
   "controllers": null,
   "element-id": "SSSS",
   "element-uri": "/api/virtualization-hosts/57ab94c8-03e6-11e1-baf3-001f163805d8/
    virtual-switches/SSSS",
   "ip-timeout": 5,
   "is-connect-uplinks": true,
   "is-use-any-available-controller": true,
   "layer-mode": "eth",
   "mtu-size": null,
   "mtu-size-enforcement": "external",
   "name": "SSSS",
   "parent": "/api/virtualization-hosts/57ab94c8-03e6-11e1-baf3-001f163805d8",
   "queue-size": 8,
   "real-uplinks": [],
   "router": null,
   "switch-status": "Defined",
   "type": "iedn"
}
```

Figure 83. Get Virtual Switch Properties: Response for virtual switch of type "iedn"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 06:01:38 GMT
content-type: application/json;charset=UTF-8
content-length: 718
{
   "bridge-connection-status": null,
   "bridge-device-number": null,
   "bridge-value-type": null,
   "class": "virtual-switch",
   "controllers": null,
   "element-id": "TESTQPOP",
   "element-uri": "/api/virtualization-hosts/57ab94c8-03e6-11e1-baf3-001f163805d8/
    virtual-switches/TESTQPOP",
   "ip-timeout": 5,
   "is-connect-uplinks": true,
   "is-gvrp-enabled": false,
   "is-use-any-available-controller": true,
   "is-vlan-aware": true,
   "layer-mode": "eth",
   "mtu-size": null,
   "mtu-size-enforcement": "external",
   "name": "TESTQPOP",
   "parent": "/api/virtualization-hosts/57ab94c8-03e6-11e1-baf3-001f163805d8",
   "queue-size": 8,
   "real-uplinks": [],
   "router": null,
   "switch-status": "Defined",
   "type": "qdio",
   "vlan-id": "AWARE",
   "vlan-native-id": "1",
   "vlan-object-uri": "",
   "vlan-port-type": "ACCESS"
}
```

Figure 84. Get Virtual Switch Properties: Response for virtual switch of type "qdio"

Create IEDN Virtual Switch

The **Create IEDN Virtual Switch** operation creates an IEDN virtual network switch for the z/VM Virtualization Host.

HTTP method and URI

POST /api/virtualization-host/{*virt-host-id*}/virtual-switches/operations/create-iedn

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`name`	String	Required	The **name** property of iedn-virtual-switch object
`layer-mode`	String Enum	Optional	The **layer-mode** property of iedn-virtual-switch object. Default value is **"eth"**.
`router`	String Enum	Optional	The **router** property of iedn-virtual-switch object. Default value is **"none"**.

Field name	Type	Rqd/Opt	Description
queue-size	Integer	Optional	The **queue-size** property of iedn-virtual-switch object. Default value is 8.
ip-timeout	Integer	Optional	The **ip-timeout** property of iedn-virtual-switch object. Default value is 5.
is-connect-uplinks	Boolean	Optional	The **is-connect-uplinks** property of iedn-virtual-switch object. Default value is false.
real-uplinks	Array of real-uplink objects	Required if **is-connect-uplinks** is true	The **real-uplinks** property of iedn-virtual-switch object. Default value is an empty list. If **is-connect-uplinks** is true, the list is required and contains 1-3 real-uplink objects.
is-use-any-available-controller	Boolean	Optional	The **is-use-any-available-controller** property of iedn-virtual-switch object. Default value is true. If the value is true and virtual switch is successfully created and there is a controller available in the z/VM, this property will be set to false, and the available controller will be used and shown in the **controllers** property.
controllers	Array of Strings	Required if **is-use-any-available-controller** is false	The **controllers** property of iedn-virtual-switch object
bridge-value-type	String	Optional; Allowed only if **bridge-device-number** has been specified	The **bridge-value-type** property of the iedn-virtual-switch object. Default value is **"secondary"**.
bridge-device-number	String	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true and the**layer-mode** property of the virtual switch is **"eth"**	The **bridge-device-number** property of the iedn-virtual-switch object. Default value is null.
bridge-connection-status	String	Optional; Allowed only if **bridge-device-number** has been specified	Values: • **"connect"** - Connect the bridge port • **"disconnect"** - Disconnect the bridge port Default value is **"connect"**.
mtu-size-enforcement	String	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true	The **mtu-size-enforcement** property of the iedn-virtual-switch object.Default value is **"external"**.
mtu-size	Integer	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true and the**mtu-size-enforcement** property of the virtual switch is **"user-defined"**	The **mtu-size** property of the iedn-virtual-switch object. Required if **mtu-size-enforcement** is **"user-defined"**.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	The **element-uri** property of the created virtual-switch element.

Description

This operation creates the IEDN virtual switch for the identified Virtualization Host and then returns its URI. The response also includes a **Location** header that provides this URI.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*
- Action/task permission to the **Manage Virtual Switches** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents."

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	81	A virtual switch with the **name** specified in the request body already exists on the Virtualization Host with object-id *{virt-host-id}*.
	87	The iedn-virtual-switch object **bridge-device-number** property specified is already in use.
	88	The iedn-virtual-switch object **bridge-device-number** property specified is not defined available to the z/VM Virtualization Host.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	80	The Virtualization Host already has a vSwitch with a bridge where the iedn-virtual-switch object **bridge-value-type** property is **"primary"**.
	81	The Virtualization Host already has four vSwitches with a bridge where the iedn-virtual-switch object **bridge-value-type** property is **"secondary"**. Four is the maximum.

HTTP error status code	Reason code	Description
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Create QDIO Virtual Switch

The **Create QDIO Virtual Switch** operation creates a QDIO virtual network switch for the z/VM Virtualization Host.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/virtual-switches/operations/create-qdio

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
name	String	Required	The **name** property of qdio-virtual-switch object
is-vlan-aware	Boolean	Optional	The **is-vlan-aware** property of qdio-virtual-switch object. Default value is false.
vlan-id	String	Optional; Allowed only if **is-vlan-aware** is true	The **vlan-id** property of qdio-virtual-switch object. Default value is **""**.
vlan-port-type	String Enum	Optional; Allowed only if **is-vlan-aware** is true	The **vlan-port-type** property of qdio-virtual-switch object. Default value is **"access"**.
vlan-native-id	String	Optional; Allowed only if **is-vlan-aware** is true	The **vlan-native-id** property of qdio-virtual-switch object. Default value is **"1"**.
is-gvrp-enabled	Boolean	Optional; Allowed only if **is-vlan-aware** is true	The **is-gvrp-enabled** property of qdio-virtual-switch object. Default value is false.
layer-mode	String Enum	Optional	The **layer-mode** property of qdio-virtual-switch object. Default value is **"eth"**.
router	String Enum	Optional	The **router** property of qdio-virtual-switch object. Default value is **"none"**.
queue-size	Integer	Optional	The **queue-size** property of qdio-virtual-switch object. Default value is 8.

Field name	Type	Rqd/Opt	Description
ip-timeout	Integer	Optional	The **ip-timeout** property of qdio-virtual-switch object. Default value is 5.
is-connect-uplinks	Boolean	Optional	The **is-connect-uplinks** property of qdio-virtual-switch object. Default value is false.
real-uplinks	Array of real-uplink objects	Required if **is-connect-uplinks** is true	The **real-uplinks** property of qdio-virtual-switch object. Default value is an empty list. If **is-connect-uplinks** is true, the list is required and contains 1-3 real-uplink objects.
is-use-any-available-controller	Boolean	Optional	The **is-use-any-available-controller** property of qdio-virtual-switch object. Default value is true. If the value is true and virtual switch is successfully created and there is a controller available in the z/VM, this property will be set to false, and the available controller will be used and shown in the **controllers** property
controllers	Array of Strings	Required if **is-use-any-available-controller** is false	The **controllers** property of qdio-virtual-switch object
bridge-value-type	String	Optional; Allowed only if **bridge-device-number** has been specified	The **bridge-value-type** property of the qdio-virtual-switch object. Default value is **"secondary"**.
bridge-device-number	String	Optional; Allowed only if the**is-bridge-capable** property of the virtualization host is true and the**layer-mode** property of the virtual switch is **"eth"**	The **bridge-device-number** property of the qdio-virtual-switch object. Default value is null.
bridge-connection-status	String	Optional; Allowed only if **bridge-device-number** has been specified	Values: • **"connect"** - Connect the bridge port. (Default value) • **"disconnect"** - Disconnect the bridge port Default value is **"connect"**.
mtu-size-enforcement	String	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true	The **mtu-size-enforcement** property the qdio-virtual-switch object. Default value is **"external"**.
mtu-size	Integer	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true and the**mtu-size-enforcement** property of the virtual switch is **"user-defined"**	The **mtu-size** property the qdio-virtual-switch object. Required if **mtu-size-enforcement** is **"user-defined"**.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	The **element-uri** property of the created virtual-switch element.

Description

This operation creates the QDIO virtual switch for the identified Virtualization Host and then returns its element URI. The response also includes a **Location** header that provides this URI.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*
- Action/task permission to the **Manage Virtual Switches** task

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents."

If errors occur, the following HTTP status codes are provided, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	81	A virtual switch with the **name** specified in the request body already exists on the Virtualization Host with object-id *{virt-host-id}*.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	80	The Virtualization Host already has a vSwitch with a bridge where the qdio-virtual-switch object **bridge-value-type** property is **"primary"**.
	81	The Virtualization Host already has four vSwitches with a bridge where the qdio-virtual-switch object **bridge-value-type** property is **"secondary"**. Four is the maximum.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Get Switch Controllers

The **Get Switch Controllers** operation gets a list of controllers for the z/VM Virtualization Host. Controllers are z/VM TCP/IP virtual machines used to manage OSA-Express devices associated with virtual switches. For details, see the *z/VM CP Commands and Utility Reference*, SC24-6175.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/operations/get-switch-controllers

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
controllers	Array of controller objects	An array of controller object as defined in the next table.

Each nested controller object provides the properties for a single switch controller, and has the following format:

Field name	Type	Description
name	String (0-8)	Name for the controller
is-available	Boolean	True if the controller is available to control an additional set of OSA-Express devices associated with the virtual switch.
vdev-range	String (1-9)	Identifies the device range where the OSA-Express devices associated with a virtual switch can be attached. The value is two virtual device address values separated by a hyphen (e.g. “8800-88FF”) or “*”, which indicates that the virtual device address used to attach the OSA-Express devices is the same as the real device address identified by the virtual switch's real-uplinks.
is-ip	Boolean	True if the virtual switch controller can initialize an OSA-Express device in IP mode. See Virtual Switch layer mode.
is-eth	Boolean	True if the virtual switch controller can initialize an OSA-Express device in ETH mode. See Virtual Switch layer mode.
is-vlan-arp	Boolean	True if the virtual switch controller can register IP addresses on the OSA-Express with the proper VLAN groups (VLAN_ARP).
is-gvrp	Boolean	True if the virtual switch controller can register VLAN IDs in use on a virtual switch with GVRP-aware switches (GVRP).
is-linkagg	Boolean	True if the controller can control virtual switches that are using link aggregation. A Link Aggregation port group is two or more links that are grouped together to appear as a single logical link.
is-isolation	Boolean	True if the controller can control virtual switches that are using the isolation setting.

Description

This operation lists the controllers for the identified Virtualization Host.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.
- Action/task permission to the **Manage Virtual Switches** task

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the "Response body contents" on page 195.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3/operations/get-switch-controllers HTTP/1.1
x-api-session: lkcjo7eteu67cygzoknowww8caid5jiyck6yfupyqz3619t3r
```

Figure 85. Get Switch Controllers: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 05:49:49 GMT
content-type: application/json;charset=UTF-8
content-length: 653
{
   "controllers": [
      {
         "is-available": true,
         "is-eth": true,
         "is-gvrp": true,
         "is-ip": true,
         "is-isolation": true,
         "is-linkagg": true,
         "is-vlan-arp": true,
         "name": "DTCVSW1",
         "vdev-range": "0600-F000"
      },
      {
         "is-available": true,
         "is-eth": true,
         "is-gvrp": true,
         "is-ip": true,
         "is-isolation": true,
         "is-linkagg": true,
         "is-vlan-arp": true,
         "name": "DTCENS1",
         "vdev-range": "*"
      }
   ]
}
```

Figure 86. Get Switch Controllers: Response

Update Virtual Switch

The **Update Virtual Switch** operation modifies an existing virtual network switch for the z/VM Virtualization Host.

HTTP method and URI

POST /api/virtualization-host/{*virt-host-id*}/virtual-switches/{*virtual-switch-id*}

URI variables

Variable	Description
{virt-host-id}	Object ID of the Virtualization Host
{virtual-switch-id}	Element ID of the virtual switch

Request body contents

For IEDN virtual switch:

Field name	Type	Rqd/Opt	Description
`router`	String Enum	Optional; Allowed only if **layer-mode** is **"ip"**	The **router** property of iedn-virtual-switch object
`queue-size`	Integer	Optional	The **queue-size** property of iedn-virtual-switch object.
`ip-timeout`	Integer	Optional	The **ip-timeout** property of iedn-virtual-switch object.
`is-connect-uplinks`	Boolean	Optional	The **is-connect-uplinks** property of iedn-virtual-switch object
`is-use-any-available-controller`	Boolean	Optional	The **is-use-any-available-controller** property of iedn-virtual-switch object. If the value is true and virtual switch is successfully updated and there is a controller available in the z/VM, this property will be set to false, and the available controller will be used and shown in the **controllers** property.
`controllers`	Array of Strings	Optional; Allowed only if **is-use-any-available-controller** is false	The **controllers** property of iedn-virtual-switch object.
`real-uplinks`	Array of real-uplink objects	Required if **is-connect-uplinks** is true	The **real-uplinks** property of iedn-virtual-switch object. If **is-connect-uplinks** is true, the list is required and contains 1-3 real-uplink objects.
`bridge-value-type`	String	Optional; Allowed only if **bridge-device-number** has been specified	The **bridge-value-type** property of the iedn-virtual-switch object
`bridge-device-number`	String	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true and the **layer-mode** property of the virtual switch is **"eth"**	The **bridge-device-number** property of the iedn-virtual-switch object
`bridge-connection-status`	String	Optional; Allowed only if **bridge-device-number** has been specified	Values: • **"connect"** - Connect the bridge port. (Default value) • **"disconnect"** - Disconnect the bridge port
`mtu-size-enforcement`	String	Optional; Allowed only if the**is-bridge-capable** property of the virtualization host is true	The **mtu-size-enforcement** property of the iedn-virtual-switch object

Field name	Type	Rqd/Opt	Description
mtu-size	Integer	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true and the **mtu-size-enforcement** property of the virtual switch is **"user-defined"**	The **mtu-size** property of the iedn-virtual-switch object. Required if **mtu-size-enforcement** is **"user-defined"**.

For QDIO virtual switch:

Field name	Type	Rqd/Opt	Description
router	String Enum	Optional; Allowed only if **layer-mode** is **"ip"**	The **router** property of the qdio-virtual-switch object
queue-size	Integer	Optional	The **queue-size** property of qdio-virtual-switch object.
ip-timeout	Integer	Optional	The **ip-timeout** property of qdio-virtual-switch object.
is-connect-uplinks	Boolean	Optional	The **is-connect-uplinks** property of qdio-virtual-switch object
is-use-any-available-controller	Boolean	Optional	The **is-use-any-available-controller** property of qdio-virtual-switch object. If the value is true and virtual switch is successfully updated and there is a controller available in the z/VM, this property will be set to false, and the available controller will be used and shown in the **controllers** property.
controllers	Array of Strings	Optional; Allowed only if **is-use-any-available-controller** is false	The **controllers** property of qdio-virtual-switch object.
real-uplinks	Array of real-uplink objects	Required if **is-connect-uplinks** is true	The **real-uplinks** property of qdio-virtual-switch object. If **is-connect-uplinks** is true, the list is required and contains 1-3 real-uplink objects.
bridge-value-type	String	Optional; Allowed only if **bridge-device-number** has been specified	The **bridge-value-type** property of the qdio-virtual-switch object
bridge-device-number	String	Optional; Allowed only if the**is-bridge-capable** property of the virtualization host is true and the **layer-mode** property of the virtual switch is **"eth"**	The **bridge-device-number** property of the qdio-virtual-switch object

Field name	Type	Rqd/Opt	Description
`bridge-connection-status`	String	Optional; Allowed only if **bridge-device-number** has been specified	Values: • **"connect"** - Connect the bridge port. (Default value) • **"disconnect"** - Disconnect the bridge port
`mtu-size-enforcement`	String	Optional; Allowed only if the **is-bridge-capable** property of the virtualization host is true	The **mtu-size-enforcement** property of the qdio-virtual-switch object
`mtu-size`	Integer	Optional; Allowed only if the**is-bridge-capable** property of the virtualization host is true and the **mtu-size-enforcement** property of the virtual switch is **"user-defined"**	The **mtu-size** property of the qdio-virtual-switch object. Required if **mtu-size-enforcement** is **"user-defined"**.

Description

This operation modifies the virtual switch for the identified virtual server.

Authorization requirements

This operation has the following authorization requirements:
- Action/task role permission to the **Manage Virtual Switches** task
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	87	The iedn-virtual-switch object **bridge-device-number** property specified is already in use.
	88	The iedn-virtual-switch object **bridge-device-number** property specified is not defined available to the z/VM Virtualization Host.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
409 (Conflict)	80	The Virtualization Host} already has a vSwitch with a bridge where the qdio-virtual-switch object **bridge-value-type** property is **"primary"**.
	81	The Virtualization Host already has four vSwitches with a bridge where the qdio-virtual-switch object **bridge-value-type** property is **"secondary"**. Four is the maximum.
	82	The **bridge-connection-status** property for an iedn-virtual-switch or qdio-virtual-switch object must be **"disconnected"** to update bridge properties.
	83	To change the **bridge-device-number** property for an iedn-virtual-switch or qdio-virtual-switch object, it must first be set to **"none"**. This removes the bridge device.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Delete Virtual Switch

The **Delete Virtual Switch** operation deletes an existing virtual network switch specified by an element identifier for the z/VM Virtualization Host with the given object identifier.

HTTP method and URI

DELETE /api/virtualization-hosts/{*virt-host-id*}/virtual-switches/{*virtual-switch-id*}

URI variables

Variable	Description
{virt-host-id}	Object ID of the Virtualization Host
{virtual-switch-id}	Element ID of the virtual switch

Description

This operation deletes the virtual switch for the identified Virtualization Host.

Authorization requirements

This operation has the following authorization requirements:
- Action/task role permission to the **Manage Virtual Switches** task
- Object access permission to hosting-environment of the Virtualization Host with object-id *{virt-host-id}*.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
403 (Forbidden)	0	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Activating a Virtualization Host

This operation is not directly accessible; it will occur as a side effect of activating the hosting environment. If **auto-start-virtual-servers** is true, the Virtualization Host activation will also activate all virtual servers on the Virtualization Host whose **auto-start** property is true.

See the Activate sections of the hosting environment objects for operation details, including URI parameters, response body contents, authorization requirements, and HTTP status and reason codes.

Asynchronous result description

Once the activation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the Activate Virtualization Host request.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields job-status-code and job-reason-code) which are set as indicated in operation description in “Job Status and Reason Codes” on page 203. The results also include a job-results nested object that has the following form:

Field name	Type	Description
failed-virtual-servers	Array of String/URI	Array of the virtual servers whose activation failed. This field only exists if the virtualization host activated, but one or more virtual servers failed to activate, indicated by job-status-code 500 and job-reason-code 102.

Job Status and Reason Codes

Job status code	Job reason code	Description
200 (OK)	N/A	Activation completed successfully.
500 (Server Error)	100	Virtualization host activation failed.
	101	Virtualization host activation job timed out.
	102	Virtualization host activation succeeded, but some virtual servers failed to activate, see the response body for a list of virtual servers that failed to activate.

Deactivating a Virtualization Host

This operation is not directly accessible; it will occur as a side effect of deactivating the hosting environment.

See the Deactivate sections of the hosting environment objects for operation details, including URI parameters, response body contents, authorization requirements, and HTTP status and reason codes.

Asynchronous result description

Once the deactivation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the Deactivate Virtualization Host request.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields job-status-code and job-reason-code) which are set as indicated in operation description below. The results also include a job-results nested object that has the following form:

Field name	Type	Description
failed-virtual-servers	Array of String/URI	Array of the virtual servers whose deactivation failed. This field only exists if the virtualization host deactivated, but one or more virtual servers failed to deactivate, indicated by job-status-code 500 and job-reason-code 102.

Job Status and Reason Codes

Job status code	Job reason code	Description
200 (OK)	N/A	Deactivation completed successfully.
500 (Server Error)	100	Virtualization host deactivation failed.
	101	Virtualization host deactivation job timed out.
	102	Virtualization host deactivation succeeded, but some virtual servers failed to deactivate, see the response body for a list of virtual servers that failed to deactivate.

SMAPI Error Response Body

If an operation encounters an error while communicating with a z/VM Virtualization Host via SMAPI, a 503 (Service Unavailable) status code is returned with reason code 100. If an operation is able to

communicate with a z/VM Virtualization Host and execute a command via SMAPI, but the SMAPI command returns a non-zero return code, a 503 (Service Unavailable) status code is returned with reason code 101 and the standard error response body is extended with the following information about the SMAPI command failure:

Field name	Type	Description
smapi-command	String	The name of the SMAPI command that encountered the error
smapi-return-code	String	The return code returned when executing the SMAPI command
smapi-reason-code	String	The reason code returned when executing the SMAPI command
smapi-message	String	The error message returned when executing the SMAPI command or **"null"** if no error message was provided

Inventory Service Data

Information about the Virtualization Hosts managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for Virtualization Host objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by inventory class, implicitly via a containing category, or by default) that objects of the various Virtualization Host type-specific inventory classes are to be included. An entry for a particular Virtualization Host is included only if the API user has object-access permission to that object and the applicable type-specific inventory class has been specified, as described in the following table:

Inventory class	Includes Virtualization Hosts with "type" value
`power-vm-virtualization-host`	power-vm
`prsm-virtualization-host`	prsm
`x-hyp-virtualization-host`	x-hyp
`zvm-virtualization-host`	zvm

For each Virtualization Host object to be included, the inventory response array includes the following:

- An entry that is a JSON object with the same contents as is specified in the Response Body Contents section for the **Get Virtualization Host Properties** operation. That is, the data provided is the same as would be provided if a **Get Virtualization Host Properties** operation were requested targeting this object.
- An array entry for each Virtual Switch object associated with the virtualization host. For each such resource, an entry is included that is a JSON object with the same contents as specified in the Response Body Contents section of the Get Virtual Switch Properties operation. As a result, the data provided is the same as would be obtained if a Get Virtual Switch Properties operation where requested for each resource listed by a List Virtual Switches operation targeting the virtualization host.
- An array entry for each Virtualization Host Storage Resource object associated with the virtualization host. For each such resource, an entry is included that is a JSON object with the same contents as specified in the Response Body Contents section of the **Get Virtualization Host Storage Resource Properties** operation, however storage path accessibility status is not provided. (More specifically, the accessible property of path-information-fcp and path-information-eckd nested objects will always null.) This data is described in Chapter 11, "Storage Management," on page 295. As a result, the data provided is the same as would be obtained if a **Get Virtualization Host Storage Resource Properties** operation were requested with the **include-path-accessibility** query parameter specified as false for each resource listed by a List Virtualization Host Storage Resources operation targeting the virtualization host.
- An array entry for each Virtualization Host Storage Group object associated with the virtualization host. For each such group, an entry is included that is a JSON object with the same contents as specified in the Response Body Contents section of the **Get Virtualization Host Storage Group**

Properties operation. This data is described in Chapter 11, "Storage Management," on page 295. As a result, the data provided is the same as would be obtained if a **Get Virtualization Host Storage Group Properties** operation where requested for each group listed by a **List Virtualization Host Storage Groups** operation targeting the virtualization host.

The array entry for a virtualization host object will appear in the results array before entries for associated virtual switches, virtualization host storage resources, or groups.

Sample inventory data

The following fragments are examples of the JSON objects that would be included in the Get Inventory response to describe a single Virtualization Host object of a particular type. These objects would appear as array entries in the response array.

```
{
      "acceptable-status": [
         "operating"
      ],
      "auto-start-virtual-servers": false,
      "class": "virtualization-host",
      "description": "",
      "has-unacceptable-status": "true",
      "minimum-memory-size-for-virtual-server": 256,
      "name": "B.1.14",
      "object-id": "baab1cd2-2990-11e0-8d5b-001f163803de",
      "object-uri": "/api/virtualization-hosts/baab1cd2-2990-11e0-8d5b-001f163803de",
      "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
     be79e49dd340",
      "status": "operating",
      "type": "power-vm",
      "virtual-server-shutdown-timeout": 300
   }
```

Figure 87. Virtualization host object: Sample inventory data for a virtualization host of type "power-vm"

```
{
      "acceptable-status": [
         "operating",
         "channel-acceptable"
      ],
      "auto-start-virtual-servers": false,
      "class": "virtualization-host",
      "description": "Initial description",
      "has-unacceptable-status": "false",
      "hosting-environment": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
      "is-locked": false,
      "name": "R32",
      "object-id": "bab76208-2990-11e0-8d5b-001f163803de",
      "object-uri": "/api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de",
      "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
     be79e49dd340",
      "status": "operating",
      "type": "prsm",
      "virtual-server-shutdown-timeout": 200
   }
```

Figure 88. Virtualization host object: Sample inventory data for a virtualization host of type "prsm"

```
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start-virtual-servers": true,
   "class": "virtualization-host",
   "description": "",
   "has-unacceptable-status": "true",
   "hosting-environment": "/api/blades/b8210bc0-2d1e-11e0-ae81-e41f13fe1430",
   "is-locked": false,
   "maximum-allowed-ide-devices": 3,
   "maximum-allowed-virtual-processors": 16.0,
   "maximum-memory-size-for-virtual-server": 125829,
   "memory-increment-in-megabytes": 1,
   "minimum-memory-size-for-virtual-server": 1,
   "mixed-mode-boot-restriction": true,
   "name": "B.1.03",
   "object-id": "931b25d6-82e1-11e0-b9e4-f0def10bff8d",
   "object-uri": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/nodes/37c6f8a9-8d5e-3e5d-8466-
  be79e49dd340",
   "status": "operating",
   "type": "x-hyp",
   "virtual-server-shutdown-timeout": 302
}
```

Figure 89. Virtualization host object: Sample inventory data for a virtualization host of type "x-hyp"

Virtual Server Object

A virtual server object represents a single zEnterprise virtual server.

Data Model

This object includes the properties defined the “Base managed object properties schema” on page 33, with the following class-specific specialization:

Table 41. Virtual server object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
name	(w)(pc)	String (1-64)	The display name of the virtual server. Names must be unique to other existing virtual servers on the virtualization host. The format of the name varies based on the virtual server type: • **"zvm"**: 1-8 characters, characters may be uppercase alphanumeric or any of the following characters: “@#$+:” • **"power-vm"**, **"x-hyp"**: 1-64 characters, must begin with an alphabetic characters, other characters may be characters may be alphanumeric, a space, or any of the following characters: “!@#$%^*()_+-=,.:;'~” • **"prsm"**: 1-8 characters, alphanumeric For virtual servers of type **"prsm"**, this property is the LPAR name as defined in the active IOCDS and is immutable. This property is also immutable for virtual servers whose **type** property is **"zvm"**, though it is user-defined through the Create Virtual Server operation. For **"power-vm"** and **"x-hyp"** virtual servers, this property may only be modified when the virtual server's status is **"not-operating"**.
description	(w)(pc)	String	Read-only for virtual servers of type **"prsm"**.
object-uri	—	String/ URI	The canonical URI path for a virtual server object is of the form `/api/virtual-servers/{object-id}`.
parent	—	String/ URI	The URI path of the virtualization host that hosts this virtual server.
class	—	String (14)	Always **"virtual-server"**.

Table 41. Virtual server object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
status	(sc)	String Enum	The current operational status of the managed resource. **"prsm"** values: • **"not-activated"** - indicates the virtual server's LPAR image was not activated. • **"operating"** - indicates all CPs are operating • **"not-operating"** - indicates no CPs are operating • **"not-communicating"** - indicates the HMC is not communicating with the support element • **"exceptions"** - indicates the virtual server has a problem of some sort, such as not being able to access storage • **"status-check"** - indicates at least one CP is operating, but at least one CP is not operating. When a **"prsm"** virtual server has this status, its LPAR Image is not available. **"power-vm"** and **"x-hyp"** values: • **"operating"** - indicates virtual server is in an activated and running state • **"not-operating"** - indicates the virtual server is deactivated, in the process of loading, or has a error • **"not-communicating"** - indicates the HMC is not communicating with the support element • **"exceptions"** - indicates the virtual server has a problem of some sort, such as not being able to access storage • **"status-check"** - indicates the virtual server is powered on, but the virtualization host is not communicating so there is no status available • **"migrating"** - indicates the virtual server is in the process of migrating to another virtualization host • **"starting"** - indicates the virtual server in the process of powering on • **"stopping"** - indicates the virtual server in the process of powering off. **"zvm"** values: • **"operating"** – indicates the virtual machine is logged on and has no current failure conditions • **"not-operating"**– indicates the virtual machine is logged on, but currently has some sort of failure condition • **"not-activated"** – indicates the virtual machine is not logged on • **"not-communicating"** - indicates the HMC is not communicating with the support element • **"logoff-timeout-started"** – indicates a logoff timeout has been started for the virtual machine • **"storage-limit-exceeded"** – indicates the storage limit has been exceeded for the virtual machine • **"forced-sleep"** – indicates the virtual machine is logged on, but has been placed into a forced sleep state • **"unknown"** – indicates the status of the virtual machine reported by z/VM was not recognized.

Table 41. Virtual server object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
acceptable-status	(w)(pc)	Array of String Enum	For virtual servers of type **"prsm"**, this property is immutable. For **"zvm"** virtual servers, the following status values may not be set as acceptable: **"not-communicating"** and **"unknown"**.
additional-status	(sc)	String Enum	The property is not provided for virtual servers of type **"power-vm"**, **"x-hyp"**, or **"prsm"**. For **"zvm"** virtual servers, this property only applies when the status is **"not-operating"** and only if the status is **"not-operating"** because of the current state of its hosting-environment. In other cases for a **"zvm"** virtual server, the value of this property is null. Values when non-null: • **"host-env-not-activated"** - indicates the **"zvm"** virtual server is not operating because the hosting-environment is not activated • **"host-env-not-capable"** - indicates the **"zvm"** virtual server is not operating because the hosting-environment is not allowing communications. This can be due to a temporary condition at the support element for the associated CPC or can be due to SMAPI communication problems. • **"host-env-not-operating"** – indicates the **"zvm"** virtual server is not operating because the hosting-environment is not operating

Data model notes

For an x Hyp virtual server, the UUID assigned by zManager as the virtual server's object ID and thus provided as the virtual server's **object-id** property is also used as the virtual server's System Management BIOS (SMBIOS) UUID. As a result, this value is visible to software running within the virtual server as the UUID field of the SMBIOS System Identification (Type 1) structure. Guest operating system interfaces or utilities, such as the Linux **lspci** command, may be available to query this value. Because the object ID is available both from outside the guest as well as within, it can serve as a reliable correlating value for management application that have a need to associate data obtained from these two environments.

Class specific additional properties

In addition to the properties defined in included schemas, this object includes the following additional class-specific properties.

Note: Many properties are only valid for virtual servers of specific “type”. These value are only included in a virtual server object if the virtual server is of that type. For example a virtual server with type **"power-vm"** will define a **processing-mode** property (**"power-vm"** only) and **mac-prefix** property (**"power-vm"**, **"zvm"**) but not an **initial-share-mode** property (**"zvm"** only).

Other properties are only valid when mutable prerequisite properties have specific values. When such properties are not valid, their value is null. For instance a **"power-vm"** virtual server's **initial-virtual-processors** property value is null when the **processing-mode** value is **"dedicated"**.

Table 42. Virtual server object: class specific additional properties

Name	Qualifier	Type	Description	Supported "type" values
type	—	String Enum	Type of the virtual server. Values: • **"power-vm"** - a virtual server that has been defined on an IBM POWER7® blade • **"x-hyp"** - a virtual server that has been defined on an IBM System x blade • **"zvm"** - a virtual server that has been defined on an IBM z/VM operating system instance that is participating as an ensemble-managed Virtualization Host • **"prsm"** - the virtual server representation of an LPAR image.	All
gpmp-status	(pc)	String Enum	Status of the System z Guest Platform Management Provider (GPMP). Values: • **"unknown"** – Guest performance agent status could not be determined • **"not-operating"** – Guest performance agent is not operating • **"operating"** – Guest performance agent is operating • **"status-check"** – There is a failure in communications between the virtual server and the guest performance agent. If virtual server type is **"prsm"**, **gpmp-status** is only available if virtual server is running z/OS®.	All
cpu-perf-mgmt-enabled	(w)(pc)	Boolean	If true, management of processor performance is enabled for this virtual server if management of processor performance is enabled at the ensemble level. **Note:** Management of processor performance at the ensemble level is enabled by virtual server type. See the ensemble object's **power-vm-cpu-perf-mgmt-enabled** and **zvm-cpu-perf-mgmt-enabled** properties.	power-vm, zvm
hostname	(pc)	String	Virtual server host name. This data is only available if the guest platform management provider is running on the virtual server.	power-vm, x-hyp, zvm
os-name	(pc)	String	The name given to this system by its operating system. This data is only available if the guest platform management provider is running on the virtual server.	All
os-type	(pc)	String	The type of operating system that is running on the virtual server. This data is only available if the guest platform management provider is running on the virtual server.	All
os-level	(pc)	String	The release level of the operating system, as reported by the OS itself. This data is only available if the guest platform management provider is running on the virtual server.	All

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
mac-prefix	(pc)	mac-prefix Object	MAC address provides the means of identification that forwards frames within the LAN segment. This prefix is the first part of all MAC addresses assigned for the virtual server. The remaining bits are dynamically assigned when the virtualization host has to generate a MAC address for a network adapter.	power-vm, zvm
processing-mode[1]	(w)	String Enum	The manner in which virtual processors are associated with the physical processors available on the virtualization host. Values: • **"shared"** - In shared mode, the virtual servers can use fractions of physical processors. The processor capacity is assigned in 0.1 units of physical processor, equivalent to 1.0 virtual processing unit. The virtual processor units can be shared among multiple virtual servers. • **"dedicated"** - In dedicated mode, the processors are assigned in whole units of physical processors. The processors that are dedicated to the virtual server cannot be used by other virtual servers. The virtual server cannot use any processors other than its own dedicated processors.	power-vm
minimum-dedicated-processors[1]	(w)	Integer	Defines the minimum number of dedicated processors that the virtual server can use.[3] Prerequisite: **processing-mode** is **"dedicated"**. Limits: • >= 1 • <= **maximum-allowed-dedicated-processors**	power-vm
initial-dedicated-processors[1, 2]	(w)(pc)	Integer	Defines the initial number of dedicated processors that the virtual server can use; the number of dedicated processors to be provided to the virtual server when it is next activated.[3] Prerequisite: **processing-mode** is **"dedicated"**. **"power-vm"** Limits: • >= 1 • <= **maximum-allowed-dedicated-processors** • >= **minimum-dedicated-processors**	power-vm
maximum-dedicated-processors[1]	(w)	Integer	The upper limit for the number of dedicated processors to for the virtual server to consume.[3] Prerequisite: **processing-mode** is **"dedicated"** **"power-vm"** Limits: • >= 1 • <= **maximum-allowed-dedicated-processors** • >= **minimum-dedicated-processors** • >= **initial-dedicated-processors**	power-vm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
minimum-virtual-processors[1]	(w)	Integer	Defines the minimum number of virtual processors that the virtual server can use.[3] Prerequisite: **processing-mode** is **"shared"**. Limits: • >= 1 • <= 10 * **minimum-processing-units** • >= **minimum-processing-units**	power-vm
initial-virtual-processors[1, 2] ([1]applies to PowerVM and x Hyp only, [2]applies only to PowerVM)	(w)	Integer	Defines the initial number of virtual processors that the virtual server can use; the number of virtual processors to be provided to the virtual server when it is next activated.[3] Prerequisites: • **"x-hyp"**, **"zvm"**: none • **"power-vm"**: **processing-mode** is **"shared"** **"power-vm"** limits: • >= 1 • <= 10 * **initial-processing-units** • >= **initial-processing-units** • >= **minimum-virtual-processors** **"x-hyp"** limits: • >= 1 • <= **maximum-allowed-virtual-processors** **"zvm"** limits: • >= 1	power-vm, x-hyp, zvm
maximum-virtual-processors[1] ([1]applies to PowerVM only)	(w)	Integer	The upper limit for the number of virtual processors to for the virtual server to consume.[3] Prerequisites: • **"x-hyp"**, **"zvm"**: none • **"power-vm"**: **processing-mode** is **"shared"** **"power-vm"** limits: • >= 1 • <= 10 * **maximum-processing-units** • >= **maximum-processing-units** • >= **minimum-virtual-processors** • >= **initial-virtual-processors** • <= **maximum-allowed-virtual-processors** **"zvm"** limits: • >= 1 • >= **initial-virtual-processors** • <= **maximum-allowed-virtual-processors**	power-vm, zvm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
minimum-processing-units[1]	(w)	Float	The minimum number of processing units required for this virtual server to start running. Prerequisite: **processing-mode** is **"shared"**. Limits: • >= 0.1 * **minimum-virtual-processors** • <= **minimum-virtual-processors** • <= **maximum-allowed-processing-units** • Fixed to two decimal places	power-vm
initial-processing-units[1, 2]	(w)	Float	The number of processing units representing the initial processor scheduling target for this virtual server. If resources are available, the virtual server may receive more than this amount.[4] Prerequisite: **processing-mode** is **"shared"**. Limits: • >= .1 * **initial-virtual-processors** • <= **initial-virtual-processors** • <= **maximum-allowed-processing-units** • >= **minimum-processing-units** • Fixed to two decimal places	power-vm
maximum-processing-units[1]	(w)	Float	The maximum number of processing units that will be allocated to the virtual server (processor utilization capping). Prerequisite: **processing-mode** is **"shared"**. If you have enabled processor management, Maximum processing units defines the upper limit for zManager. A virtual server with a capacity equal to this maximum value cannot receive resources from other virtual servers on the blade. Limits: • >= 0.1 * **maximum-virtual-processors** • <= **maximum-virtual-processors** • <= **maximum-allowed-processing-units** • >= **minimum-processing-units** • >= **initial-processing-units** • Fixed to two decimal places	power-vm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
initial-share-mode	(w)	String Enum	Defines the virtual servers' initial share of system resources either relative to other virtual servers or absolutely. Values: • **"relative"** - Grants all virtual servers different priorities for processor and I/O. A relative share allocates to a virtual server a portion of the total system resources minus those resources allocated to virtual servers with an absolute share. Also, a virtual server with a relative share receives access to system resources that is proportional with respect to other virtual servers with relative shares. For example, if a virtual server (VM1) has a relative share of 100, and a second virtual server (VM2) has a relative share of 200, VM2 receives twice as much access to system resources as VM1. • **"absolute"** - Grants universal access and priority over all other virtual servers. An absolute share allocates to a virtual server an absolute percentage of all available system resources. For example, if you assign a virtual server an absolute share of 50%, CP allocates to that virtual server approximately 50% of all available resources (regardless of the number of other virtual servers running).	zvm
initial-shares	(w)(pc)	Float	If **initial-share-mode** is **"relative"**, defines the initial share value for the virtual server relative to other virtual servers on the virtualization host. When maximum-share-mode is **"relative"**, this value must be an Integer between 1 and min (maximum-shares, 10000). If **initial-share-mode** is **"absolute"**, defines the initial share value for the virtual server absolutely. When maximum-share-mode is **"absolute"**, this value must be a Number between 0.1 and min (maximum-shares, 100), fixed to one decimal place.	zvm
share-limit	(w)	String	Define how the virtual server's share of system resources is limited. Values: • **"none"** - Specifies that a server share of processing resource is not limited. • **"soft"** - Specifies that the share of processing resource is limited but at times these servers can receive more than their limit, if no other server can use the available resources. • **"hard"** - Specifies that the share of processing resource is limited. These servers can not receive more than their limit.	zvm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
maximum-share-mode	(w)	String Enum	Defines the virtual servers' maximum share of system resources either relative to other virtual servers or absolutely. Prerequisites: **share-limit** is **"soft"** or **"hard"**. Values: • **"relative"** - Grants all virtual servers different priorities for processor and I/O. A relative share allocates to a virtual server a portion of the total system resources minus those resources allocated to virtual servers with an absolute share. Also, a virtual server with a relative share receives access to system resources that is proportional with respect to other virtual servers with relative shares. For example, if a virtual server (VM1) has a relative share of 100, and a second virtual server (VM2) has a relative share of 200, VM2 receives twice as much access to system resources as VM1. • **"absolute"** - Grants universal access and priority over all other virtual servers. An absolute share allocates to a virtual server an absolute percentage of all available system resources. For example, if you assign a virtual server an absolute share of 50%, CP allocates to that virtual server approximately 50% of all available resources (regardless of the number of other virtual servers running).	zvm
maximum-shares	(w)	Float	If **maximum-share-mode** is **"relative"**, defines the maximum share value for the virtual server relative to other virtual servers on the virtualization host. When **maximum-share-mode** is **"relative"**, this value must be an Integer between 1 and min (maximum-shares, 10000). If **maximum-share-mode** is **"absolute"**, defines the maximum share value for the virtual server absolutely. When **maximum-share-mode** is **"absolute"**, this value must be a Number between 0.1 and min (maximum-shares, 100), fixed to one decimal place. Prerequisites: **share-limit** is **"soft"** or **"hard"**.	zvm
minimum-memory[1]	(w)	Integer	Minimum memory value for use by the virtual server, specified in megabytes (MB). Must be a multiple of virtualization host's **memory-increment-in-mega-bytes** value. **"power-vm"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes**	power-vm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
initial-memory[1, 2] ([1]applies to PowerVM and x Hyp only, [2]applies only to PowerVM)	(w)(pc)	Integer	Initial memory value for use by the virtual server, specified in MB. Must be a multiple of virtualization host's **memory-increment-in-mega-bytes** value. **"power-vm"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes** • >= **minimum-memory** • <= **maximum-memory** • <= **maximum-memory-size-for-virtual-server** **"x-hyp"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes** • <= **maximum-memory-size-for-virtual-server** **"zvm"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes** • <= **maximum-memory** • <= **maximum-memory-size-for-virtual-server**	power-vm, x-hyp, zvm
maximum-memory[1] ([1]applies to PowerVM only)	(w)	Integer	Maximum memory value for use by the virtual server, specified in MB. Must be a multiple of virtualization host's **memory-increment-in-mega-bytes** value. **"power-vm"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes** • >= **minimum-memory** • >= **initial-memory** • <= **maximum-memory-size-for-virtual-server** **"zvm"** limits: • >= **minimum-memory-size-for-virtual-server** • multiple of **memory-increment-in-mega-bytes** • <= **maximum-memory-size-for-virtual-server**	power-vm, zvm
workloads	(c)(pc)	Array of String/ URI	The canonical URI path of each workload resource group to which the virtual server is assigned.	All
network-adapters	—	Array of objects	Array of nested Network Adapter objects defining the virtual server's network adapters. The nested objects are of type network-adapter-power, network-adapter-x-hyp, network-adapter-zvm or network-adapter-prsm depending on the type of the virtual server.	All
virtual-disks	—	Array of objects	Array of nested Virtual Disk objects defining the virtual server's virtual disks.	power-vm, x-hyp, zvm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
mounted-media-name	—	String (0-255)	The display name of the mounted ISO or null if no media is mounted.	power-vm, x-hyp
boot-mode	(w)(pc)	String Enum	The boot mode of the virtual server. Values: • **"normal"** - The virtual server boots in normal mode. • **"sms"** - The virtual server boots to the System Management Services (SMS) menu. • **"diagnostic-default-boot-list"**- The sequence of devices read at startup. • **"diagnostic-stored-boot-list"** - The virtual server performs a service mode boot using the service mode boot list saved in NVRAM. • **"open-firmware-prompt"** - The virtual server boots to the open firmware prompt.	power-vm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
boot-sequence	(w)(pc)	Array of String Enum	List of boot sources. Values: • **"virtual-disk"** – virtual-disks will be tried in order • **"network-adapter"** – the first network-adapter will be used as a boot source • **"virtual-media"** – virtual media will be used as a boot source List must contain one and only one element for **type "power-vm"** **"virtual-disk"** is not a valid value if the virtual server's **virtual-disks** property is empty. **"network-adapter"** is not a valid value if the virtual server's **network-adapter** property is empty. **"power-vm"** network-adapter boot notes: • If boot-network-adapter-client-ip, boot-network-adapter-subnet-ip, boot-network-adapter-gateway-ip, and boot-network-adapter-server-ip are not defined, boot network settings defined via DHCP. • If boot-network-adapter-client-ip, boot-network-adapter-subnet-ip, and boot-network-adapter-server-ip are defined, but boot-network-adapter-gateway-ip is not defined, boot network settings are manually defined and netboot is limited to its own subnet. • If boot-network-adapter-client-ip, boot-network-adapter-subnet-ip, boot-network-adapter-gateway-ip, and boot-network-adapter-server-ip are all defined, boot network settings are manually defined and netboot is not limited to its own subnet. **"x-hyp"** virtual-disk boot notes: • Booting is always enabled for Virtual DVDs (virtual media), virtual disks, and virtual network adapters if devices of these types are defined for the virtual server. Although it is possible to change the order in which booting is attempted, it is not possible to disable booting from devices of these types if they are present for the virtual server. See the text directly after this table for considerations that apply to the value of the boot-sequence property. • If the virtual server resides on a node that is being managed by Support Element Version 2.11.1, a virtual network interface (if present) must be either the first or last entry in the virtual server's boot sequence. That is, the value **"network-adapter"** must appear as either the first or last entry in the **boot-sequence** property. **"x-hyp"** virtual-disk boot notes: • If the virtualization host's mixed-mode-boot-restriction value is true and the virtual-disks property defines both **"virtio"** and **"ide"** virtual disks, the virtual server will only boot from an **"ide"** virtual-disk when the **boot-sequence** value contains **"virtual-disk"**.	power-vm, x-hyp

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
boot-network-adapter-client-ip	(w)	String (0-39)	The IP address of the virtual server used during network boot in dotted-decimal form ("nnn.nnn.nnn.nnn"). Prerequisites: **boot-sequence** contains **"network-adapter"**.	power-vm
boot-network-adapter-subnet-ip	(w)	String (0-17)	The IPv4 subnet mask of the network used during network boot of the virtual server in dotted-decimal form ("nnn.nnn.nnn.nnn"). Prerequisites: **boot-sequence** contains **"network-adapter"**.	power-vm
boot-network-adapter-gateway-ip	(w)	String (0-39)	The IP address of the gateway system used during network boot of the virtual server in dotted-decimal form ("nnn.nnn.nnn.nnn"). Prerequisites: **boot-sequence** contains **"network-adapter"**.	power-vm
boot-network-adapter-server-ip	(w)	String (0-39)	The IP address of the boot server on which the disk boot source files reside in dotted-decimal form ("nnn.nnn.nnn.nnn"). Prerequisites: **boot-sequence** contains **"network-adapter"**.	power-vm
keylock	(w)	String Enum	Indicates the state of the system key lock at boot time. Value: • **"normal"** – Enables a regular operating system boot of the virtual server including all services, under program control. • **"manual"** - Requires system operator to manually boot the virtual server. This setting is useful for service scenarios. The virtual server boots to the diagnostics menu, then you have to open a console to the virtual server and work with the displayed menu to choose the next steps.	power-vm
auto-start	(w)(pc) (w only for PowerVM and x Hyp)	Boolean	If true, this virtual server is automatically started when its hosting virtualization host is started.	All
dlpar-enabled	(w)	Boolean	If true, this virtual server is configured for Dynamic Logical Partitioning. Note: This setting does not enable any DLPAR prerequisites (such as the Resource Monitoring and Control network).	power-vm
dlpar-active	—	Boolean	If true, DLPAR support is enabled and specific virtual server properties may be updated when the virtual server status is **"operating"**.	power-vm
gpmp-support-enabled	(w)(pc) (ro for PR/SM)	Boolean	If true, guest platform management provider support is enabled, allowing the GPMP to gather performance data for work running on the virtual server. **Note:** This only defines if support is enabled, not if it is active.	All

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
gpmp-version	—	String	Guest Platform Performance Manager version or keyword **"unavailable"** if not known. If virtual server type is **"prsm"**, **"gpmp-version"** is only available if virtual server is running z/OS.	power-vm, x-hyp, zvm
password	(w)	String (1-8)	The password or passphrase to be used for authentication. Password must meet the guidelines defined by the VM system. 1-8 uppercase characters long.	zvm
privilege-classes	(w)	String (1-32)	String defining z/VM CP privilege classes denoted by the letters A through Z (uppercase), the numbers 1 through 6, and the word "ANY". 1-32 characters long	zvm
ipl-device	(w)(pc)	String (1-8)	Specifies the virtual device that you want to IPL. It is the virtual device number or sysname of the virtual device.[5] A virtual device number is a four character hex number. The sysname is a 1 - 8 uppercase alphanumeric-character name of the named saved system you want to IPL.	zvm
ipl-load-parameters	(w)(pc)	String (0-8)	The z/VM LOADPARM option value; used to pass a load parameter of up to 8 bytes of data to the operating system you are IPLing.[5] 0-8 characters long, character may be uppercase alphanumeric, a space, or a period	zvm
ipl-parameters	(w)(pc)	String (0-64)	String defining the z/VM IPL parameter list.[5] 0:64 characters.	zvm
associated-logical-partition	—	String/URI	The canonical URI path for the Logical Partition that is the PR/SM virtual server.	prsm
inband-monitoring-enabled	(w)(pc)	Boolean	If true, in-band monitoring support is enabled, allowing the hypervisor to gather performance metrics for the virtual server. Prerequisites: parent virtualization host's **inband-monitoring-supported** value is true.	power-vm, x-hyp

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
keyboard-language	(w)(pc)	String	String describing the locale required by graphical consoles connecting to the virtual server. The value is **null** if the parent virtualization host's **supported-keyboard-languages** array is **null** or empty. String value is the locale's language, country, and variant values separated by underscores: • The language value is either an empty string or a lowercase ISO 639 language code (2-4 characters). • The country value is either an empty string or an uppercase ISO 3166 two-letter code. • The variant value is either an empty string or a string defining variations for a language/country pair. For example, Serbian locales have Cyrillic ("Cyrl") and Latin ("Latn") variants. If the language is an empty string, the string will begin with an underscore. If both the language and country fields are empty strings, the value will be an empty string. Example values: "en", "en_GB", "en_US", "sr_RS", "sr_RS_Cyrl", and "sr_RS_Latn". Value must be defined in the parent virtualization host's **supported-keyboard-languages** value array.	x-hyp
perf-policies	(pc)	Array of Objects	The list of workloads and performance policies that are active on the virtual server, each with the activation status of the policy on the virtual server. The list will contain one element for each workload with a performance policy that has been activated on the virtual server in the form of a Virtual Server Performance Policy Nested Object, as described in Table 43 on page 222.	All
power-vm-partition-id	—	Integer	The identifier of the PowerVM partition in which the virtual server is running. A virtual server is assigned to a PowerVM partition during the virtual server activation process, and may be assigned to a different partition each time it is activated. Therefore, this property has a value only when the virtual server's **status** property is **"operating"**. If the virtual server is in some other status, the value of this property is null.	power-vm

Table 42. Virtual server object: class specific additional properties (continued)

Name	Qualifier	Type	Description	Supported "type" values
Notes: [1] These properties represent configuration values for the virtual server that are used to establish the virtual server's initial configuration when it is activated. [2] These properties update runtime values when the virtual server status is **"operating"** and DLPAR is active (dlpar-active value is true). If DLPAR is not active, updating these properties will not affect runtime values. DLPAR support is only available for **"power-vm"**. [3] Note: Because zManager-managed **"power-vm"** virtual servers consume resources on the Power ASB only when activated, it is possible to overcommit and define a set of virtual servers on a virtualization host that together require more virtual processors than can be simultaneously supported. PowerVM resource limits are enforced at the time that a virtual server is activated. [4] This value is a configuration value that is used to set the scheduling goal when the virtual server is activated. If CPU performance management is active for this virtual server, the run time value may be adjusted as the virtual server operates. This will not change the "initial" value and thus does not trigger property change event. The current processing unit goal for the virtual server will be available as a virtual server metric. [5] The maximum length of the IPL Directory statement resulting from the ipl-device, ipl-load-parameters, and ipl-parameters values may not exceed 72 characters. Refer to the *z/VM CP Planning and Administration Guide* for details on the IPL Directory statement.				

The following considerations apply to the value of the **boot-sequence** property:

- Because a virtual server always has a virtual DVD device, the value **"virtual-media"** must always appear somewhere within the value of the **boot-sequence** property.
- If the virtual server has one or more virtual disks defined, then the value **"virtual-disk"** must appear within **boot-sequence** property. When the first virtual disk is defined for a virtual server, the **boot-sequence** property is updated to add **"virtual-disk"** as the last entry in the list. When the last virtual disk is removed, the value **"virtual-disk"** is removed from the **boot-sequence** property.
- Similarly, if the virtual server has one or more virtual network interfaces defined, then the value **"network-adapter"** must also appear within the **boot-sequence** property. When the first virtual network interface is defined for a virtual server, the **boot-sequence** property is updated to add **"network-adapter"** as the last entry in the list. When the last virtual network interface is removed, the value **"network-adapter"** is removed from the **boot-sequence** property.

The Virtual Server Performance Policy Nested Object is nested within a virtual server object to encapsulate the performance policies that are active on the virtual server and their activation status with respect to the virtual server.

Table 43. Virtual Server Performance Policy Nested Object

Name	Type	Description
`workload-uri`	String/URI	The canonical URI path of the workload resource group to which the performance policy identified by policy-uri is assigned.
`policy-name`	String	The **name** property of the Performance Policy object.
`activation-status`	String/URI	The status of activating the performance policy on the virtual server. Possible values are: • **"initializing"** - Status is initializing (not yet known). • **"successful"** - The policy has been successfully activated on the virtual server. • **"failed"**- Policy activation failed. • **"pending"** - Policy activation is pending (in progress).

mac-prefix object: A mac-prefix object defines a **"power-vm"** or **"zvm"** virtual server's mac-prefix property value.

Table 44. mac-prefix object properties

Name	Type	Description
mac-address	String (17)	The MAC address represented as 6 groups of two lower-case hexadecimal digits separated by colons (:), e.g. “01:23:45:67:89:ab”. Length is 17 characters. The MAC address uses the ensemble prefix.
prefix-length	Integer	The bit length of the MAC address prefix. This is a 2-digit value with these parameters in the range 12-44.

network-adapter objects: The bit length of the MAC address prefix. This is a 2-digit value with these parameters in the range 12-44.

The network-adapter-power object defines a network adapter of a virtual-server of type **"power-vm"**.

Table 45. network-adapter-power object properties

Name	Qualifier	Type	Description
element-id	—	String	Unique ID for the virtual network adapter within the scope of the containing virtual server. It is a randomly generated integer value when the object is created. Currently this value will change if this object is modified or the group of network adapters of the virtual server is reordered.
element-uri	—	String/ URI	The canonical URI path for the virtual network adapter is of the form `/api/virtual-servers/{virtual-server-id}/network-adapters/{element-id}`, where *{virtual-server-id}* is the **object-id** of the virtual server.
network-uri	(w)	String/ URI	The canonical URI path for the associated virtual network or **null** if the network adapter is not connected to a virtual network.
mac-address	—	String (17)	The MAC address of the network adapter represented as 6 groups of two lowercase hexadecimal digits separated by colons (:), e.g. “01:23:45:67:89:ab”. Length is 17 characters. The MAC address uses the ensemble prefix.

The network-adapter-x-hyp object defines a network adapter of a virtual-server object of type **"x-hyp"**.

Table 46. network-adapter-x-hyp object properties

Name	Qualifiers	Type	Description
element-id	—	String	Unique ID for the virtual network adapter within the scope of the containing virtual server.
element-uri	—	String/ URI	The canonical URI path for the virtual network adapter is of the form `/api/virtual-servers/{virtual-server-id}/network-adapters/{element-id}`, where *{virtual-server-id}* is the **object-id** of the virtual server.
emulation-mode	—	String Enum	The network adapter emulation mode. Values: • **"e1000"** – Intel E1000 • **"rtl8139"** – Realtek 8139 • **"virtio"** – RedHat virtio
network-uri	(w)	String/ URI	The canonical URI path for the associated virtual network or **null** if the network adapter is not connected to a virtual network.

Table 46. network-adapter-x-hyp object properties (continued)

Name	Qualifiers	Type	Description
mac-address	—	String (17)	The MAC address of the network adapter represented as 6 groups of two lowercase hexadecimal digits separated by colons (:), e.g. “01:23:45:67:89:ab”. Length is 17 characters.

The network-adapter-zvm object defines a network adapter of a virtual-server object of type **"zvm"**.

Note: Some properties are only valid for network adapters of specific “type”. These value are only included in a network-adapter-zvm object if the network adapter is of that type. For example, a network adapter with type **rmc** will not define an **interface-type** property.

Other properties are only valid when mutable prerequisite properties have specific values. When such properties are not valid, their value is null. For instance a network-adapter-zvm's **real-device-address** property value is null when the **interface-type** value is **"virtual-iedn"**.

Table 47. network-adapter-zvm object properties

Name	Qualifier	Type	Description
element-id	—	String (1-4)	Unique ID for the virtual network adapter within the scope of the containing virtual server. This element-id is actually the **virtual-device-address** listed in this table. This element ID is not immutable. It will be changed if the **virtual-device-address** of this object is modified.
element-uri	—	String/ URI	The canonical URI path for the virtual network adapter is of the form `/api/virtual-servers/{virtual-server-id}/network-adapters/{element-id}`, where *{virtual-server-id}* is the **object-id** of the virtual server. This URI is not immutable because the **element-id** component of it can change. See the description for the **element-id** property.
virtual-device-address	(w)	String (1-4)	Virtual device address 1-4 character hex string
device-count	—	Integer	The number of device addresses to reserve
type	(ro)	String Enum	Network adapter type. Values: • **"osx"**: OSX is a CHPID type. A network adapter that connects through an OSX CHPID to provide connectivity to a virtual network of the IEDN (Intra-ensemble Data Network). • **"osd"**: OSD is a CHPID type. A network adapter that connects through an OSD CHPID to provide connectivity to external network. • **"iqd"**: IQD is a CHPID type. A network adapter that connects through an IQD CHPID to provide connectivity to a HiperSockets™ network that is not part of the IEDN. • **"iqdx"**: IQDX is a CHPID type that provides connectivity to the IEDN HiperSockets network. • **"rmc"**: Identifies the network adapter is the Remote Monitoring and Control network adapter.

Table 47. network-adapter-zvm object properties (continued)

Name	Qualifier	Type	Description
interface-type	(w)	String Enum	Network adapter switch type. Legal values based on **type** value. osx values: • **"none"** • **"virtual-iedn"**: Virtual IEDN switch • **"physical-iedn"**: Phycsical IEDN switch osd values: • **"none"** • **"virtual-iedn"**: Virtual IEDN switch • **"virtual-qdio"**: Virtual QDIO switch • **"physical-qdio"**: Physical QDIO switch iqd values: • **"none"** • **"physical-iqdn"**: Physical IQDN switch iqdx values: • **"none"** • **"physical-iedn"**: Physical IQDX switch
real-device-address	(w)	String (1-4)	The device address that has been assigned to the port on the switch that the “dedicated” NIC is mapped to. Prerequisites: **interface-type** is **"physical-iedn"**, **"physical-qdio"**, or **"physical-iqdn"** 1-4 character hex number (range 0-FFFF).
switch-uri	(w)	String/URI	The switch to which the network adapter is connected. Prerequisites: **interface-type** is not **"none"**. • If **interface-type** is **"virtual-iedn"** or **"virtual-qdio"**, the value is the **element-uri** of the **virtual-switch** object in use by the z/VM network adapter. • If **interface-type** is **"physical-iedn"** and type is **"osx"**, the value is the **element-uri** of the **network-adapter-prsm** object in use by the z/VM network adapter.

Table 47. network-adapter-zvm object properties (continued)

Name	Qualifier	Type	Description
port-mode	(w)	String Enum	The port mode, or null if the virtual-switch to which the network adapter was connected no longer exists. Prerequisites: **interface-type** is **"virtual-iedn"** or **"virtual-qdio"**. If **interface-type** is **"virtual-qdio"**, the virtual switch identified by **switch-uri** must be VLAN aware (its **is-vlan-aware** property is true). Values: • **"trunk"**: When the switch port is configured in trunk mode, it will allow the flow of traffic from multiple virtual networks (i.e. VLANS). The port must be configured with those virtual networks. • **"access"**: When the switch port is configured in access mode, it will support a single virtual network. Traffic from the virtual server's network adapter will be tagged with the virtual network configured for this port, and traffic destined to the virtual server on this port will be verified that it is tagged with the configured virtual network.
vlan-ids	(w)	String (0-19)	A space-delimited String defining the VLAN IDs. Prerequisites: **interface-type** is **"virtual-qdio"** Value may contain one of the following: • zero to four VLAN IDs • zero to two ranges of VLAN IDs. A VLAN ID is a decimal integer from 1-4094. A range is defined by two VLAN IDs separated by a hyphen. If **port-mode** is **"access"** this string must define a single VLAN ID. Examples: “”, “0 10”, “0 10 100 2000”, “0-10 2000-2000”, “0-10 1000-2000”
network-uris	(w)	Arrays of String/URI	A list of associated Virtual Networks. Each item represents the canonical URI path for the associated virtual network. Prerequisites: **interface-type** is **"virtual-iedn"** or **"physical-iedn"**. List must contain at least one network URI. If **port-mode** is **"access"**, list must contain exactly one URI.

The network-adapter-prsm object defines a network adapter of a virtual-server object of type **"prsm"**.

Table 48. network-adapter-prsm object properties

Name	Qualifier	Type	Description
element-id	—	String	Unique ID for the virtual network adapter within the scope of the containing virtual server.
element-uri	—	String/URI	The canonical URI path for the virtual network adapter is of the form `/api/virtual-servers/{virtual-server-id}/network-adapters/{element-id}`, where *{virtual-server-id}* is the **object-id** of the virtual server.
type	—	String Enum	The physical switch type. Either **"osx"** or **"idqx"**.

Table 48. network-adapter-prsm object properties (continued)

Name	Qualifier	Type	Description
css	—	String (1)	The channel subsystem ID
chpid	—	String (2)	The channel path ID
network-uris	(w)	Array of String/ URI	A list including the canonical URI path for each associated virtual network.

Virtual disk objects: A Virtual Disk is virtual storage space provided by a virtualization host to a guest virtual server. A Virtual Disk is based upon a storage resource, but may be further virtualized by a Virtualization Host.

Every virtual disk object contains the following base properties in addition to type-specific properties:

Table 49. Virtual disk object properties

Name	Qualifier	Type	Description
element-id	—	String (36)	The unique identifier for the virtual disk instance. This identifier is in the form of a UUID.
element-uri	—	String/URI	Canonical URI path of the virtual disk object, in the form `/api/virtual-servers/{virtual-server-id}/virtual-disks/{element-id}`, where *{virtual-server-id}* is the **object-id** of the virtual server.
name	(w)	String (1-64)	The name of the virtual disk. It must consist only of alphanumeric characters, spaces and the following special characters: "._-", and it must begin with an alphabetic character.
description	(w)	String (0-1024)	The description for the virtual disk. It must consist only of alphanumeric characters, spaces and the following special characters: "._-".
type	—	String Enum	The type of virtual disk. Values: • **"fullpack"** - A virtual disk that is backed by an entire virtualization host storage resource • **"storage-group-based"** - A virtual disk that is backed by resources allocated from a virtualization host storage group. This type only applies to virtual disks of a virtual server whose **type** property is **"zvm"**. • **"linked"** - A virtual disk that is linked to a virtual disk owned by another virtual server. This type only applies to virtual disks of a virtual server whose **type** property is **"zvm"**.
size	—	Long	The size of the virtual disk, in bytes.
owner	—	String/URI	Canonical URI path of the virtual server that owns this virtual disk.
emulation-mode	(w)	String Enum	The disk I/O emulation mode. Values: • **"virtio"** – This virtual disk emulates VIRTIO • **"ide"** – This virtual disk emulates IDE Only valid for virtual disks of a virtual server whose **type** property of **"x-hyp"**.

A fullpack-virtual-disk object contains information about a fullpack virtual disk of a non-z/VM virtual server (i.e., the virtual server's **type** property is not **"zvm"**, and the virtual disk's **type** property is **"fullpack"**).

In addition to base properties, a fullpack-virtual-disk object contains the following properties:

Table 50. fullpack-virtual-disk object properties

Name	Qualifier	Type	Description
backing-virtualization-host-storage-resource	(w)	String/URI	Canonical URI path of the virtualization host storage resource object that backs this virtual disk.

A fullpack-virtual-disk-zvm object contains information about a z/VM virtual server's fullpack virtual disk (i.e., the virtual server's **type** property is **"zvm"**, and the virtual disk's **type** property is **"fullpack"**).

In addition to base properties, a fullpack-virtual-disk-zvm object contains the following properties:

Table 51. fullpack-virtual-disk-zvm object properties

Name	Qualifier	Type	Description
device-address	—	String (1-4)	The virtual device address. This is the device address by which the virtual server knows this virtual disk. The string form of a 1-4 digit hexadecimal number.
access-mode	(w)	String Enum	The access mode describing the virtual server's permission to read and/or write to this virtual disk. The values are defined in Table 54 on page 229.
read-password	(w)	String (0-8)	The read password for this virtual disk. Characters must be uppercase.
write-password	(w)	String (0-8)	The write password for this virtual disk. Characters must be uppercase.
multi-password	(w)	String (0-8)	The multiple-write password for this virtual disk. Characters must be uppercase.
backing-virtualization-host-storage-resource	—	String/URI	Canonical URI path of the virtualization host storage resource object that backs this virtual disk

A storage-group-based-virtual-disk object contains information about a virtual server's storage-group-based virtual disk (i.e., the virtual disk's **type** property is **"storage-group-based"**).

In addition to base properties, a storage-group-based-virtual-disk object contains the following properties:

Table 52. storage-group-based-virtual-disk object properties

Name	Qualifier	Type	Description
device-address	—	String (1-4)	The virtual device address. This is the device address by which the virtual server knows this virtual disk. The string form of a 1-4 digit hexadecimal number.
access-mode	(w)	String Enum	The access mode describing the virtual server's permission to read and/or write to this virtual disk. The values are defined Table 54 on page 229.
read-password	(w)	String (0-8)	The read password for this virtual disk. Characters must be uppercase.
write-password	(w)	String (0-8)	The write password for this virtual disk. Characters must be uppercase.
multi–password	(w)	String (0-8)	The multiple-write password for this virtual disk. Characters must be uppercase.

Table 52. storage-group-based-virtual-disk object properties (continued)

Name	Qualifier	Type	Description
backing-storage-group	—	String/URI	Canonical URI path of the virtualization host storage group object that backs this virtual disk.

A linked-virtual-disk object contains information about a virtual server's linked virtual disk (i.e., the virtual disk's **type** property is **"linked"**).

In addition to base properties, a linked-virtual-disk object contains the following properties:

Table 53. linked-virtual-disk object properties

Name	Qualifier	Type	Description
device-address	—	String (1-4)	The virtual device address. This is the device address by which the virtual server knows this virtual disk. The string form of a 1-4 digit hexadecimal number.
access-mode	(w)	String Enum	The access mode describing the virtual server's permission to read and/or write to this virtual disk. The values are defined in Table 54.
base-virtual-disk	—	String/URI	Canonical URI path of the virtual disk to which this virtual disk is ultimately linked.
source-virtual-disk	—	String/URI	Canonical URI path of the virtual disk to which this virtual disk is directly linked.

The possible access modes for a virtual disk of a z/VM virtual server are listed in the following table. This table is effectively a String enumeration of the valid values for the **access-modes** property of a virtual disk object.

Table 54. Valid values for the access-modes property of a virtual disk object

Type	Description
String Enum	The virtual disk's access mode. These modes correspond exactly to the disk access modes in the z/VM user directory. Values: • **"read-only"** • **"read-write"** • **"multi-write"** • **"unsupported"** – any access mode other than the supported ones listed here. A virtual disk created by some means other than zManager-provided functions could be created with such an access mode.

Usage notes for virtual disks:

- For virtual disk objects whose **type** property is **"linked"**, the “source” virtual disk is always the same as the “base” virtual disk. The base virtual disk property is provided for the case where a virtual disk is a link to a virtual disk which is itself a link to some other virtual disk. As such a configuration is not supported by zManager, base and source will be identical.

Operations

If a virtual server operation accesses a z/VM virtualization host and encounters an error while communicating with the virtualization host via SMAPI, the response body is a SMAPI Error Response Body.

List Virtual Servers of an Ensemble

The **List Virtual Servers of an Ensemble** operation lists the virtual servers managed by the ensemble with the given identifier.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/virtual-servers

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.
type	String Enum	Optional	Filter string to limit returned objects to those that have a matching **type** property. Value must be a valid virtual server **type** property value.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`virtual-servers`	Array of objects	Array of virtual-server-info objects, described in the next table. Returned array may be empty.

Each nested virtual-server-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	The canonical URI path of the Virtual Server object
`name`	String	The **name** property of the Virtual Server object
`type`	String Enum	The **type** property of the Virtual Server object
`status`	String Enum	The **status** property of the Virtual Server object

Description

This operation lists the virtual servers that are managed by the identified ensemble. The **object-uri**, **object-id**, **name**, **type**, and **status** are provided for each.

If the object-id *{ensemble-id}* does not identify an ensemble object to which the API user has object-access permission, a 404 status code is returned.

If the **name** query parameter is specified, the returned list is limited to those virtual servers that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

If the **type** query parameter is specified, the parameter is validated to ensure it is a valid virtual server **type** property value. If the value is not valid, a 400 (Bad Request) is returned. If the value is valid, the

returned list is limited to those virtual servers that have a **type** property matching the specified value. If the **type** parameter is omitted, this filtering is not done.

A virtual server is included in the list only if the API user has object-access permission for that object. If an HMC is a manager of a virtual server but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the ensemble does not manage any virtual servers or if no virtual servers are to be included in the results due to filtering, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the ensemble whose object-id is *{ensemble-id}*
- Object access permission to each virtual server object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 230.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	105	A **type** query parameter defines an invalid value.
404 (Not Found)	1	An ensemble with object-id *{ensemble-id}* does not exist on HMC or API user does not have object-access permission for it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/virtual-servers HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
```

Figure 90. List Virtual Servers of an Ensemble: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Sat, 12 Nov 2011 21:09:23 GMT
content-type: application/json;charset=UTF-8
content-length: 1008
{
   "virtual-servers": [
      {
         "name": "APIVM2",
         "object-uri": "/api/virtual-servers/63911688-03f4-11e1-881f-001f163805d8",
         "status": "operating",
         "type": "prsm"
      },
      {
         "name": "SS-Web-Svr-1",
         "object-uri": "/api/virtual-servers/7ba96572-0d72-11e1-892f-f0def14b63af",
         "status": "not-operating",
         "type": "x-hyp"
      },
      {
         "name": "SSWSVR3",
         "object-uri": "/api/virtual-servers/86819708-0d72-11e1-bf89-f0def14b63af",
         "status": "not-activated",
         "type": "zvm"
      },
      {
         "name": "ZOS",
         "object-uri": "/api/virtual-servers/636768f6-03f4-11e1-881f-001f163805d8",
         "status": "operating",
         "type": "prsm"
      },
      {
         "name": "SS-web-Srv-2",
         "object-uri": "/api/virtual-servers/960529e2-0d3b-11e1-9f64-f0def14b63af",
         "status": "operating",
         "type": "power-vm"
      }
   ]
}
```

Figure 91. List Virtual Servers of an Ensemble: Response

List Virtual Servers of a CPC

The **List Virtual Servers of a CPC** operation lists the virtual servers managed by the CPC with the given identifier.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/virtual-servers

In this request, the URI variable *{cpc-id}* is the object ID of the CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.

Name	Type	Rqd/Opt	Description
type	String Enum	Optional	Filter string to limit returned objects to those that have a matching **type** property. Value must be a valid virtual server **type** property value.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`virtual-servers`	Array of objects	Array of nested virtual-server-info objects, described in the next table.

Each nested virtual-server-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	The canonical URI path of the Virtual Server object
`name`	String	The **name** property of the Virtual Server object
`type`	String Enum	The **type** property of the Virtual Server object
`status`	String Enum	The **status** property of the Virtual Server object

Description

This operation lists the virtual servers that are managed by the identified CPC. The **object-uri**, **name**, **type**, and **status** are provided for each.

If the object-id *{cpc-id}* does not identify a CPC object to which the API user has object-access permission or if the CPC is not a member of an ensemble, a 404 status code is returned.

If the **name** query parameter is specified, the returned list is limited to those virtual servers that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

If the **type** query parameter is specified, the parameter is validated to ensure it is a valid virtual server **type** property value. If the value is not valid, a 400 (Bad Request) is returned. If the value is valid, the returned list is limited to those virtual servers that have a **type** property matching the specified value. If the **type** parameter is omitted, this filtering is not done.

A virtual server is included in the list only if the API user has object-access permission for that object. If an HMC is a manager of a virtual server but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the CPC does not manage any virtual servers or if no virtual servers are to be included in the results due to filtering, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the cpc whose object ID is *{cpc-id}*
- Object access permission to each virtual server object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 233.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	105	A **type** query parameter defines an invalid value.
404 (Not Found)	1	A CPC with object-id *{cpc-id}* does not exist on HMC or API user does not have object-access permission for it.
	100	The CPC with object-id *{cpc-id}* is not a member of an ensemble.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/virtual-servers HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
```

Figure 92. List Virtual Servers of a CPC: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Sat, 12 Nov 2011 21:09:23 GMT
content-type: application/json;charset=UTF-8
content-length: 423
{
   "virtual-servers": [
      {
         "name": "ZOS",
         "object-uri": "/api/virtual-servers/636768f6-03f4-11e1-881f-001f163805d8",
         "status": "operating",
         "type": "prsm"
      },
      {
         "name": "SS-web-Srv-2",
         "object-uri": "/api/virtual-servers/960529e2-0d3b-11e1-9f64-f0def14b63af",
         "status": "operating",
         "type": "power-vm"
      }
   ]
}
```

Figure 93. List Virtual Servers of a CPC: Response

List Virtual Servers of a Virtualization Host

The **List Virtual Servers of a Virtualization Host** operation lists the virtual servers managed by the virtualization host with the given identifier.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/virtual-servers

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`virtual-servers`	Array of objects	Array of nested virtual-server-info objects, described in the next table.

Each nested virtual-server-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	The **object-uri** property of the Virtual Server object.

Field name	Type	Description
`name`	String	The **name** property of the Virtual Server object
`type`	String Enum	The **type** property of the Virtual Server object
`status`	String Enum	The **status** property of the Virtual Server object

Description

This operation lists the virtual servers that are managed by the identified virtualization host. The **object-uri**, **name**, **type**, and status are provided for each.

If the object-id *{virt-host-id}* does not identify a virtualization host object for which the API user has object-access permission to its hosting-environment, a 404 status code is returned.

If the **name** query parameter is specified, the returned list is limited to those virtual servers that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

A virtual server is included in the list only if the API user has object-access permission for that object. If an HMC is a manager of a virtual server but the API user does not have permission to it, that object is simply omitted from the list but no error status code results.

If the virtualization host does not manage any virtual servers or if no virtual servers are to be included in the results due to filtering, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to hosting-environment of the virtualization host with object-id *{virt-host-id}*
- Object access permission to each virtual server object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 235.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	105	A **type** query parameter defines an invalid value.
404 (Not Found)	1	A Virtualization Host with object-id *{virt-host-id}* does not exist on HMC or API user does not have object-access permission for its hosting-environment.
503 (Service Unavailable	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/71822c16-0401-11e1-8eda-001f163805d8/virtual-servers HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
```

Figure 94. List Virtual Servers of a Virtualization Host: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Sat, 12 Nov 2011 21:08:38 GMT
content-type: application/json;charset=UTF-8
content-length: 434
{
   "virtual-servers": [
      {
         "name": "SS-Web-Svr-1",
         "object-uri": "/api/virtual-servers/7ba96572-0d72-11e1-892f-f0def14b63af",
         "status": "operating",
         "type": "x-hyp"
      },
      {
         "name": "SS-FTP-Svr-1",
         "object-uri": "/api/virtual-servers/5483e6b2-09fc-11e1-9f08-001f163805d8",
         "status": "not-operating",
         "type": "x-hyp"
      }
   ]
}
```

Figure 95. List Virtual Servers of a Virtualization Host: Response

Create Virtual Server

The **Create Virtual Server** operation creates a **"power-vm"**, **"x-hyp"**, or **"zvm"** virtual server with the given properties on the identified Virtualization Host. This operation is not supported for **"prsm"** Virtualization Hosts/virtual servers.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/virtual-servers

In this request, the URI variable *{virt-host-id}* is the object ID of the Virtualization Host object.

Request body contents

Properties are only valid if they are supported for a virtual server whose **type** property matches the given **type** property value. For instance a virtual server with type **"power-vm"** will define a **processing-mode** property (PowerVM only) and **mac-prefix** property (PowerVM, z/VM) but not an **initial-share-mode** property (z/VM only).

Properties may also only be valid if prerequisite properties have specific values. For example a PowerVM virtual server's initial-virtual-processors is only valid when the **processing-mode** value is **"shared"**.

See the “Data Model” on page 206 for a complete definition of all properties including for what virtual server types they are valid and their prerequisites.

Field name	Type	Rqd/Opt	Description
`type`	String Enum	Required	The value to be set as the virtual server's **type** property. The value may not be **"prsm"**.
`name`	String	Required	The value to be set as the virtual server's **name** property.
`description`	String	Optional	The value to be set as the virtual server's **description** property. Default value: <blank>
`cpu-perf-mgmt-enabled`	Boolean	Optional	The value to be set as the virtual server's **cpu-perf-mgmt-enabled** property. Default value: true. Valid only when the ensemble's **management-enablement-level** is **"automate"**.
`processing-mode`	String Enum	Optional	The value to be set as the virtual server's **processing-mode** property. Default value: shared
`minimum-virtual-processors`	Integer	Optional	The value to be set as the virtual server's **minimum-virtual-processors** property. Default value: 1
`initial-virtual-processors`	Integer	Optional	The value to be set as the virtual server's **initial-virtual-processors** property. Default value: 1
`maximum-virtual-processors`	Integer	Optional	The value to be set as the virtual server's **maximum-virtual-processors** property. Default value based on **type**: • power-vm: max (**maximum-allowed-dedicated-processors**, **initial-virtual-processors**) • zvm: 1
`minimum-dedicated-processors`	Integer	Optional	The value to be set as the virtual server's **minimum-dedicated-processors** property. Default value: 1
`initial-dedicated-processors`	Integer	Optional	The value to be set as the virtual server's **initial-dedicated-processors** property. Default value: 1
`maximum-dedicated-processors`	Integer	Optional	The value to be set as the virtual server's **maximum-dedicated-processors** property. Default value: **maximum-allowed-dedicated-processors**
`minimum-processing-units`	Number	Optional	The value to be set as the virtual server's **minimum-processing-units** property. Default value: 0.1
`initial-processing-units`	Number	Optional	The value to be set as the virtual server's **initial-processing-units** property. Default value: 0.1
`maximum-processing-units`	Number	Optional	The value to be set as the virtual server's **maximum-processing-units** property. Default value: min (**maximum-virtual-processors**, **maximum-allowed-processing-units**)
`initial-share-mode`	String Enum	Optional	The value to be set as the virtual server's **initial-share-mode** property. Default value: **"relative"**
`initial-shares`	Number	Optional	The value to be set as the virtual server's **initial-shares** property. Default value based on **initial-share-mode**: • relative: 100 • absolute: 10.0
`share-limit`	String Enum	Optional	The value to be set as the virtual server's **share-limit** property. Default value: **"none"**
`maximum-share-mode`	String Enum	Optional	The value to be set as the virtual server's **maximum-share-mode** property. Default value: **"relative"**
`maximum-shares`	Number	Optional	The value to be set as the virtual server's **maximum-shares** property. Default value based on **maximum-share-mode**: • relative: 100 • absolute: 10.0

Field name	Type	Rqd/Opt	Description
minimum-memory	Integer	Optional	The value to be set as the virtual server's **minimum-memory** property. Default value: max (256, **initial-memory**)
initial-memory	Integer	Optional	The value to be set as the virtual server's **initial-memory** property. Default value: 1024
maximum-memory	Integer	Optional	The value to be set as the virtual server's **maximum-memory** property. Default value: **initial-memory**
boot-mode	String Enum	Optional	The value to be set as the virtual server's **boot-mode** property. Default: normal
book-network-adapter-client-ip	String	Optional	The value to be set as the virtual server's **boot-network-adapter-client-ip** property. Default value: <blank>
boot-network-adapter-subnet-ip	String	Optional	The value to be set as the virtual server's **boot-network-adapter-subnet-ip** property. Default value: <blank>
boot-network-adapter-gateway-ip	String	Optional	The value to be set as the virtual server's **boot-network-adapter-gateway-ip** property. Default value: <blank>
boot-network-adapter-server-ip	String	Optional	The value to be set as the virtual server's **boot-network-adapter-server-ip** property. Default value: <blank>
keylock	String Enum	Optional	The value to be set as the virtual server's **keylock** property. Default value: **"normal"**
auto-start	Boolean	Optional	The value to be set as the virtual server's **auto-start** property. Default value: false. Prerequisite: **type** is **"power-vm"** or **"x-hyp"**.
dlpar-enabled	Boolean	Optional	The value to be set as the virtual server's **dlpar-enabled** property. Default value: false
gpmp-support-enabled	Boolean	Optional	The value to be set as the virtual server's **gpmp-support-enabled** property. Default value: false. Prerequisite: **type** is **"power-vm"**, **"zvm"** or **"x-hyp"**.
rmc-virtual-device-address	String	Optional	The virtual device address of the resource monitoring and control network to create in order to support the System z Guest Platform Management Provider (GPMP). Prerequisite: **type** is **"zvm"** and **gpmp-support-enabled** is true. Required if **gpmp-support-enabled** is true. 1-4 character hex string
password	String	Required if **type** is **"zvm"**	The value to be set as the virtual server's **password** property. Prerequisite: **type** is **"zvm"**
privilege-classes	String	Optional	The value to be set as the virtual server's **privilege-classes** property. Default value “G”
ipl-device	String	Optional	The value to be set as the virtual server's **ipl-device** property. Default value “CMS”
ipl-load–parameters	String	Optional	The value to be set as the virtual server's **ipl-load-parameters** property. Default value <blank>
ipl-parameters	String	Optional	The value to be set as the virtual server's **ipl-parameters** property. Default value “AUTOCR”.
inband-monitoring-enabled	Boolean	Optional	The value to be set as the virtual server's **inband-monitoring-enabled** property. Default value: false
keyboard-language	String	Optional	The value to be set as the virtual server's **keyboard-language** property. Default value: the first value in the virtualization host's **supported-keyboard-languages** array or null if that array is null or empty.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`object-uri`	String/ URI	The **object-uri** property of the created Virtual Server object.

Description

This operation creates a virtual server with the values specified on the identified Virtualization Host and then returns its object-uri in the response body. The response also includes a **Location** header that provides this URI.

Any properties identified as “Required” must be included in the request body. Any properties identified as “Optional” may be excluded from the request body; if an optional property is not found in the request body, its value will be set to its default value.

If the API user does not have authority to perform the Create Virtual Server task or the zManager management enablement level for the ensemble does not allow the setting of a provided property, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virt-host-id}* does not identify a Virtualization Host object for which the API user has object-access permission to its hosting-environment. If the Virtualization Host is of **type "zvm"** and its status is not-activated, a 409 (Conflict) status code is returned.

If the Request Body Contents fail to validate, a 400 (Bad Request) status code is returned. This may occur because the document fails to define a required property or defines a property that is not supported for the given virtual server type. This may also occur if the document fails to define a single valid virtual server, for instance defining a property with an invalid value (e.g. an initial-memory value less than zero or a virtual server **type** of **"zvm"** when the Virtualization Host **type** is **"power-vm"**). A 400 (Bad Request) will also be returned if the request body contains a property that is not valid given the value of a prerequisite property (e.g. defining a value for **initial-dedicated-memory** when the value of **processing-mode** is **"shared"**).

If the Request Body Contents are valid, the virtual server is created on the target Virtualization Host and its properties are defined to their corresponding request body content's properties' values. If a property is omitted from the request body, its default value is used when creating the virtual server. The newly created virtual server will have empty list values defined for its **storage-adapters**. For PowerVM and x Hyp virtual servers, the **network-adapters** property will contain the definition of the network-adapter for the Default network and it will have a **boot-sequence** array that contains only **"network-adapter"**. For z/VM virtual servers, the **network-adapters** property value will be an empty list. The virtual server will also be a member of the default workload resource group.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Virtualization Host's hosting-environment
- Action/task permission to **Create Virtual Server** task.
- Action/task permission to the **Virtual Server Performance Details** task if the **gpmp-support-enabled** or **cpu-perf-mgmt-enabled** field is specified in the request body.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents" on page 240.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	101	A virtual server with the **name** specified in the request body already exists on the Virtualization Host with object-id *{virt-host-id}*.
	102	The object-id *{virt-host-id}* identifies a Virtualization Host whose **type** property value differs from the **type** defined in the request body.
	103	A "shares" (**initial-shares** or **maximum-shares**) property value type is not correct based on the value of its corresponding "share-mode" (**initial-share-mode** or **maximum-share-mode**). For example, the **initial-share-mode** is defined as **"relative"** but the **initial-shares** value is not an Integer.
403 (Forbidden)	1	API user does not have action permission to the **Create Virtual Server** task.
	100	The ensemble's **management-enablement-level** property value does not allow the updating of a provided property.
404 (Not Found)	1	A Virtualization Host with object-id *{virt-host-id}* does not exist on HMC or API user does not have object-access permission for its hosting-environment.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	101	Parent Virtualization Host has a **status** value that is not valid to perform the operation (attempted to create a virtual server of type **"zvm"** and Virtualization Host is not active).
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtualization-hosts/2f7bf364-03f8-11e1-8eda-001f163805d8/virtual-servers HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
content-type: application/json
content-length: 161
{
   "description": "Spacely Sprockets Web Store Web Server #2",
   "initial-memory": 2048,
   "initial-virtual-processors": 2,
   "name": "SS-Web-Svr-2",
   "type": "power-vm"
}
```

Figure 96. Create Virtual Server: Request for a virtual server of type "power-vm"

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/virtual-servers/7d298eb8-0d72-11e1-8c83-f0def14b63af
cache-control: no-cache
date: Sat, 12 Nov 2011 21:08:41 GMT
content-type: application/json;charset=UTF-8
content-length: 74
{
   "object-uri": "/api/virtual-servers/7d298eb8-0d72-11e1-8c83-f0def14b63af"
}
```

Figure 97. Create Virtual Server: Response for a virtual server of type "power-vm"

Delete Virtual Server

The **Delete Virtual Server** operation deletes the identified virtual server. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

DELETE /api/virtual-servers/{*virtual-server-id*}

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Description

A 409 (Conflict) status code is returned if the virtual server has a **status** other than **"not-operating"** or **"not-activated"**. A 409 (Conflict) status code is also returned if the virtual server is busy for the duration of the request.

This operation deletes the identified virtual server, which includes the following:
- The virtual server's network adapters are disconnected and deleted.
- The virtual server's virtual disks are removed from the virtual server.
- Virtual servers of type **"zvm"** are also removed from the z/VM Directory.
- The virtual server is removed from the Virtualization Host.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server
- Object access permission to the virtual server's Virtualization Host's hosting-environment.
- Action/task permission to **Delete Virtual Server** task

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
403 (Forbidden)	1	API user does not have action permission to the **Delete Virtual Server** task.
404 (Not Found)	1	A virtual server with object-id {*virtual-server-id*} does not exist on HMC or API user does not have object-access permission for it or its virtualization host's hosting-environment.
409 (Conflict)	1	Virtual server status is not valid to perform the operation (must be either **"not-operating"** or **"not-activated"**).
	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/virtual-servers/7ba96572-0d72-11e1-892f-f0def14b63af HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
```

Figure 98. Delete Virtual Server: Request

```
204 No Content
date: Sat, 12 Nov 2011 21:09:24 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 99. Delete Virtual Server: Response

Get Virtual Server Properties

The **Get Virtual Server Properties** operation retrieves the properties of a single Virtual Server object that is designated by its object-id.

HTTP method and URI

GET /api/virtual-servers/{*virtual-server-id*}

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Query parameters

Name	Type	Req/Opt	Description
`properties`	String	Optional	Identifies the properties to be returned for the Virtual Server object. The only supported value is **"common"**, which indicates that only a specific subset of properties that are quickly available and common to all types of virtual server should be returned. This properties included in this subset is specified in "Response body contents." If this query parameter is omitted, then all properties for the virtual server as defined in "Data Model" on page 206 are returned.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the virtual server object.

If the **properties** query parameter is not specified, the response body provides all of the properties for the virtual server as defined in the Data Model section above. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

If the **properties=common** query parameter is specified, the response body provides the values for only the following properties: **acceptable-status**, **auto-start**, **class**, **description**, **has-unacceptable-status**, **hostname**, **is-locked**, **name**, **object-id**, **object-uri**, **os-level**, **os-name**, **os-type**, **parent**, **status**, **type**, and **workloads**. Field names and data types in the JSON object are the same as the property names and data types defined in the data model. These properties are common to all types of virtual servers, and can be provided more quickly than the entire set of properties for the virtual server.

Description

Returns the current values of the properties for the virtual server object as defined in the "Data Model" on page 206.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the virtual server object designated by *{virtual-server-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	A Virtualization Host with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.

HTTP error status code	Reason code	Description
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

Retrieval of the full property set for a virtual server will generally require an interaction with the SE, and in some cases (for example, for z/VM virtual servers) may also require an interaction with the virtualization host. Some API applications may choose to represent virtual servers in a generic, rather than highly type-specialized, way and thus not need the full set of properties for each virtual server. The **properties=common** query parameter is provided to allow for improved performance for such applications by avoiding the processing that would be incurred in retrieving unneeded properties. The properties specified by **properties=common** are available for all virtual server types, and are generally cached on the HMC so that they can be retrieved quickly and without requiring an interaction with the SE or virtualization host.

Example HTTP interaction

```
GET /api/virtual-servers/7ba96572-0d72-11e1-892f-f0def14b63af HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
```

Figure 100. Get Virtual Server Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 17:11:03 GMT
content-type: application/json;charset=UTF-8
content-length: 3175
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start": false,
   "boot-mode": "normal",
   "boot-network-adapter-client-ip": "192.168.1.22",
   "boot-network-adapter-gateway-ip": "192.168.1.1",
   "boot-network-adapter-server-ip": "192.168.1.254",
   "boot-network-adapter-subnet-ip": "255.255.255.0",
   "boot-sequence": [
      "network-adapter"
   ],
   "class": "virtual-server",
   "cpu-perf-mgmt-enabled": true,
   "description": "Spacely Sprockets Web Store Web Server #2",
   "dlpar-active": false,
   "dlpar-enabled": false,
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": true,
   "hostname": null,
   "inband-monitoring-enabled": false,
   "initial-dedicated-processors": null,
   "initial-memory": 2048,
   "initial-processing-units": 0.20000000000000001,
   "initial-virtual-processors": 2,
   "is-locked": false,
   "keylock": "normal",
   "mac-prefix": {
      "mac-address": "02:d0:19:aa:76:00",
      "prefix-length": 40
   },
   "maximum-dedicated-processors": null,
   "maximum-memory": 2048,
   "maximum-processing-units": 7.0,
   "maximum-virtual-processors": 7,
   "minimum-dedicated-processors": null,
   "minimum-memory": 2048,
   "minimum-processing-units": 0.10000000000000001,
   "minimum-virtual-processors": 1,
   "mounted-media-name": "gpa.iso",
   "name": "SS-Web-Svr-2",
```

Figure 101. Get Virtual Server Properties: Response for virtual servers of "power-vm" (Part 1)

```
"network-adapters": [
      {
         "element-id": "596dd87c-0db7-11e1-9251-f0def14b63af",
         "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/
        network-adapters/596dd87c-0db7-11e1-9251-f0def14b63af",
         "mac-address": null,
         "network-uri": "/api/virtual-networks/f920171e-03f2-11e1-8e8e-0010184c8334"
      },
      {
         "element-id": "c4bbdcea-0fac-11e1-903e-f0def14b63af",
         "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/
        network-adapters/c4bbdcea-0fac-11e1-903e-f0def14b63af",
         "mac-address": null,
         "network-uri": "/api/virtual-networks/f3aece54-0db8-11e1-9e2c-00215e69dea0"
      }
   ],
   "object-id": "588d8c18-0db7-11e1-b1f1-f0def14b63af",
   "object-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af",
   "os-level": null,
   "os-name": null,
   "os-type": null,
   "parent": "/api/virtualization-hosts/2f7bf364-03f8-11e1-8eda-001f163805d8",
   "processing-mode": "shared",
   "status": "not-operating",
   "type": "power-vm",
   "virtual-disks": [
      {
         "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
        2f7bf364-03f8-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
        37699380-0fa9-11e1-b69e-f0def14b63af",
         "description": "Boot filesystem",
         "element-id": "c547fb4e-0fac-11e1-842b-f0def14b63af",
         "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/
        virtual-disks/c547fb4e-0fac-11e1-842b-f0def14b63af",
         "name": "boot",
         "owner": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af",
         "size": 17179869184,
         "type": "fullpack"
      },
      {
         "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
        2f7bf364-03f8-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
        b0e6c160-0fa9-11e1-b69e-f0def14b63af",
         "description": "Pysical vol for sysvg",
         "element-id": "c568a13c-0fac-11e1-903e-f0def14b63af",
         "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/
        virtual-disks/c568a13c-0fac-11e1-903e-f0def14b63af",
         "name": "pv01a",
         "owner": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af",
         "size": 8589934592,
         "type": "fullpack"
      }
   ],
   "workloads": [
      "/api/workload-resource-groups/f9fbe5fa-03f2-11e1-8e8e-0010184c8334"
   ]
}
```

Figure 102. Get Virtual Server Properties: Response for virtual servers of "power-vm" (part 2)

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 18:23:31 GMT
content-type: application/json;charset=UTF-8
content-length: 1121
{
   "acceptable-status": [
      "operating"
   ],
   "associated-logical-partition": "/api/logical-partitions/8394537b-1cc3-3607-88ad-06845b263439",
   "class": "virtual-server",
   "cpu-perf-mgmt-enabled": false,
   "description": "",
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": true,
   "is-locked": false,
   "name": "LP03    ",
   "network-adapters": [
      {
         "chpid": "42",
         "css": "0",
         "element-id": "OSX 0.42",
         "element-uri": "/api/virtual-servers/916f73c8-de39-11e0-a58e-f0def161133a/
        network-adapters/OSX%200.42",
         "network-uris": null,
         "type": "osx"
      },
      {
         "chpid": "43",
         "css": "0",
         "element-id": "OSX 0.43",
         "element-uri": "/api/virtual-servers/916f73c8-de39-11e0-a58e-f0def161133a/
        network-adapters/OSX%200.43",
         "network-uris": null,
         "type": "osx"
      }
   ],
   "object-id": "916f73c8-de39-11e0-a58e-f0def161133a",
   "object-uri": "/api/virtual-servers/916f73c8-de39-11e0-a58e-f0def161133a",
   "os-level": "6.2.0 - 1101",
   "os-name": "LP03",
   "os-type": "z/VM",
   "parent": "/api/virtualization-hosts/8e9cad8c-de39-11e0-a58e-f0def161133a",
   "status": "operating",
   "type": "prsm",
   "workloads": [
      "/api/workload-resource-groups/1fe39a72-de39-11e0-90a6-00215e67351a"
   ]
}
```

Figure 103. Get Virtual Server Properties: Response for virtual servers of type "prsm"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 16:56:19 GMT
content-type: application/json;charset=UTF-8
content-length: 2618
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start": false,
   "boot-sequence": [
      "network-adapter"
   ],
   "class": "virtual-server",
   "description": "Spacely Sprockets Web Store Web Server #1",
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": true,
   "hostname": null,
   "inband-monitoring-enabled": false,
   "initial-memory": 2048,
   "initial-virtual-processors": 1,
   "is-locked": false,
   "keyboard-language": "ar",
   "mounted-media-name": "gpa.iso",
   "name": "SS-Web-Svr-1",
   "network-adapters": [
      {
         "element-id": "b5f6a412-0faa-11e1-957c-f0def14b63af",
         "element-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
       network-adapters/b5f6a412-0faa-11e1-957c-f0def14b63af",
         "emulation-mode": "virtio",
         "mac-address": "02:ff:a6:b3:3c:c6",
         "network-uri": "/api/virtual-networks/f3aece54-0db8-11e1-9e2c-00215e69dea0"
      },
      {
         "element-id": "582b5d0e-0db7-11e1-b1f1-f0def14b63af",
         "element-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
       network-adapters/582b5d0e-0db7-11e1-b1f1-f0def14b63af",
         "emulation-mode": "e1000",
         "mac-address": "02:ff:1f:b8:e4:fa",
         "network-uri": "/api/virtual-networks/f920171e-03f2-11e1-8e8e-0010184c8334"
      }
   ],
   "object-id": "576569dc-0db7-11e1-b1f1-f0def14b63af",
   "object-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af",
   "os-level": null,
   "os-name": null,
   "os-type": null,
   "parent": "/api/virtualization-hosts/71822c16-0401-11e1-8eda-001f163805d8",
   "status": "not-operating",
   "type": "x-hyp",
```

Figure 104. Get Virtual Server Properties: Response for virtual servers of type "x-hyp" (Part 1)

```
   "virtual-disks": [
      {
         "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
        71822c16-0401-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
        efe826a0-094c-11e1-98b6-001f163805d8",
         "description": "Pysical vol for sysvg",
         "element-id": "b72419a0-0faa-11e1-957c-f0def14b63af",
         "element-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
        virtual-disks/b72419a0-0faa-11e1-957c-f0def14b63af",
         "emulation-mode": "virtio",
         "name": "pv01",
         "owner": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af",
         "size": 8589934592,
         "type": "fullpack"
      },
      {
         "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
        71822c16-0401-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
        efde1d9a-094c-11e1-a3f6-001f163805d8",
         "description": "Boot filesystem",
         "element-id": "b6ef9f22-0faa-11e1-957c-f0def14b63af",
         "element-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
        virtual-disks/b6ef9f22-0faa-11e1-957c-f0def14b63af",
         "emulation-mode": "ide",
         "name": "boot",
         "owner": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af",
         "size": 17179869184,
         "type": "fullpack"
      }
   ],
   "workloads": [
      "/api/workload-resource-groups/f9fbe5fa-03f2-11e1-8e8e-0010184c8334"
   ]
}
```

Figure 105. Get Virtual Server Properties: Response for virtual servers of type "x-hyp" (Part 2)

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 05:31:36 GMT
content-type: application/json;charset=UTF-8
content-length: 1133
{
   "acceptable-status": [
      "operating"
   ],
   "additional-status": null,
   "auto-start": false,
   "class": "virtual-server",
   "cpu-perf-mgmt-enabled": false,
   "description": "",
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": false,
   "hostname": null,
   "initial-memory": 32,
   "initial-share-mode": "relative",
   "initial-shares": 100.0,
   "initial-virtual-processors": 1,
   "ipl-device": "CMS",
   "ipl-load-parameters": "",
   "ipl-parameters": "AUTOCR",
   "is-locked": false,
   "mac-prefix": {
      "mac-address": "02:ca:0d:00:00:00",
      "prefix-length": 24
   },
   "maximum-memory": 0,
   "maximum-share-mode": null,
   "maximum-shares": null,
   "maximum-virtual-processors": 64,
   "name": "MEV1A",
   "network-adapters": [],
   "object-id": "1738f85e-1b6f-11e1-a8ab-f0def165da96",
   "object-uri": "/api/virtual-servers/1738f85e-1b6f-11e1-a8ab-f0def165da96",
   "os-level": null,
   "os-name": null,
   "os-type": null,
   "parent": "/api/virtualization-hosts/911f4808-de39-11e0-a58e-f0def161133a",
   "password": "999999",
   "privilege-classes": "BDEG",
   "share-limit": "none",
   "status": "operating",
   "type": "zvm",
   "virtual-disks": [],
   "workloads": [
      "/api/workload-resource-groups/1fe39a72-de39-11e0-90a6-00215e67351a"
   ]
}
```

Figure 106. Get Virtual Server Properties: Response for virtual servers of type "zvm"

Update Virtual Server Properties

The **Update Virtual Server Properties** operation updates one or more of the writeable properties of a Virtual Server. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

Fields are only valid if they are supported for a virtual server of the targeted type. For instance a virtual server with **type** property **"power-vm"** may define a processing-mode property (PowerVM only) and **mac-prefix** property (PowerVM, z/VM) but not an **initial-share-mode** property (z/VM only).

Properties may also only be valid if prerequisite properties have specific values. For example a PowerVM virtual server's **initial-virtual-processors** is only valid when the **processing-mode** value is **"shared"**.

All fields in the following table are optional. If a field is not included in the request body contents, its value will not be updated (unless a prerequisite field is changed, as noted in the table).

See the "Data Model" on page 206 for a complete definition of all properties including for what virtual server types they are valid and their prerequisites.

Field name	Type	Description
`name`	String	The value to be set as the virtual server's **name** property. Prerequisite: virtual server **type** is not **"zvm"**.
`description`	String	The value to be set as the virtual server's **description** property.
`acceptable-status`	Array of String Enum	The value to be set as the virtual server's **acceptable-status** property.
`cpu-perf-mgmt-enabled`	Boolean	The value to be set as the virtual server's **cpu-perf-mgmt-enabled** property. Default value: true. Valid only when the ensemble's **management-enablement-level** is **"automate"**.
`processing-mode`	String Enum	The value to be set as the virtual server's **processing-mode** property.
`minimum-virtual-processors`	Integer	The value to be set as the virtual server's **minimum-virtual-processors** property. Default value (used if **processing-mode** changes to **"shared"**): 1
`initial-virtual-processors`	Integer	The value to be set as the virtual server's **initial-virtual-processors** property. Default value (used if **processing-mode** changes to **"shared"**): 1
`maximum-virtual-processors`	Integer	The value to be set as the virtual server's **maximum-virtual-processors** property. Default value for power-vm (used if **processing-mode** changes to **"shared"**): max (**maximum-allowed-dedicated-processors**, **initial-virtual-processors**)
`minimum-dedicated-processors`	Integer	The value to be set as the virtual server's **minimum-dedicated-processors** property. Default value (used if **processing-mode** changes to **"dedicated"**): 1
`initial-dedicated-processors`	Integer	The value to be set as the virtual server's **initial-dedicated-processors** property. Default value (used if **processing-mode** changes to **"dedicated"**): 1
`maximum-dedicated-processors`	Integer	The value to be set as the virtual server's **maximum-dedicated-processors** property. Default value: **maximum-allowed-dedicated-processors** Default value (used if **processing-mode** changes to **"dedicated"**): **maximum-allowed-dedicated-processors**
`minimum-processing-units`	Number	The value to be set as the virtual server's **minimum-processing-units** property. Default value (used if **processing-mode** changes to **"shared"**): 0.1
`initial-processing-units`	Number	The value to be set as the virtual server's **initial-processing-units** property. Default value (used if **processing-mode** changes to **"shared"**): 0.1

Field name	Type	Description
maximum-processing-units	Number	The value to be set as the virtual server's **maximum-processing-units** property. Default value (used if **processing-mode** changes to **"shared"**): min (**maximum-virtual-processors**, **maximum-allowed-processing-units**)
initial-share-mode	String Enum	The value to be set as the virtual server's **initial-share-mode** property.
initial-shares	Float	The value to be set as the virtual server's **initial-shares** property. Default value based on **initial-share-mode** (used if **initial-share-mode** changes): • relative: 100 • absolute: 10.0
share-limit	String Enum	The value to be set as the virtual server's **share-limit** property.
maximum-share-mode	String Enum	The value to be set as the virtual server's **maximum-share-mode** property. Default value (used if **share-limit** changes from **"none"** to **"soft"** or **"hard"**): relative
maximum-shares	Float	The value to be set as the virtual server's **maximum-shares** property. Default value based on **maximum-share-mode** (used if **maximum-share-mode** changes): • relative: 100 • absolute: 10.0
minimum-memory	Integer	The value to be set as the virtual server's **minimum-memory** property.
initial-memory	Integer	The value to be set as the virtual server's **initial-memory** property.
maximum-memory	Integer	The value to be set as the virtual server's **maximum-memory** property.
boot-mode	String Enum	The value to be set as the virtual server's **boot-mode** property.
boot-sequence	Array of String Enum	The value to be set as the virtual server's **boot-sequence** property.
book-network-adapter-client-ip	String	The value to be set as the virtual server's **boot-network-adapter-client-ip** property. Default value (used if **boot-sequence** changes to [**"network-adapter"**]): <blank>
boot-network-adapter-subnet-ip	String	The value to be set as the virtual server's **boot-network-adapter-subnet-ip** property. Default value (used if **boot-sequence** changes to [**"network-adapter"**]): <blank>
boot-network-adapter-gateway-ip	String	The value to be set as the virtual server's **boot-network-adapter-gateway-ip** property. Default value (used if **boot-sequence** changes to **"network-adapter"**): <blank>
boot-network-adapter-server-ip	String	The value to be set as the virtual server's **boot-network-adapter-server-ip** property. Default value (used if **boot-sequence** changes to **"network-adapter"**): <blank>
keylock	String Enum	The value to be set as the virtual server's **keylock** property.
auto-start	Boolean	The value to be set as the virtual server's **auto-start** property.
dlpar-enabled	Boolean	The value to be set as the virtual server's **dlpar-enabled** property.
gpmp-support-enabled	Boolean	The value to be set as the virtual server's **gpmp-support-enabled** property.

Field name	Type	Description
rmc-virtual-device-address	String	The virtual device address of the resource monitoring and control network to create in order to support the System z Guest Platform Management Provider (GPMP). Prerequisite: **type** is **"zvm"** and **gpmp-support-enabled** is true. Required if **gpmp-support-enabled** changes to true. 1-4 character hex string
password	String	The value to be set as the virtual server's **password** property.
privilege-classes	String	The value to be set as the virtual server's **privilege-classes** property.
ipl-device	String	The value to be set as the virtual server's **ipl-device** property.
ipl-load–parameters	String	The value to be set as the virtual server's **ipl-load-parameters** property.
ipl-parameters	String	The value to be set as the virtual server's **ipl-parameters** property.
inband-monitoring-enabled	Boolean	The value to be set as the virtual server's **inband-monitoring-enabled** property.
keyboard-language	String	The value to be set as the virtual server's **keyboard-language** property.

Description

This operation updates a virtual server's properties with the values specified on the identified Virtualization Host.

If the API user does not have access action permission to Virtual Server Details or the zManager management enablement level for the ensemble does not allow the setting of a provided property, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is also returned if the object-id *{virt-host-id}* does not identify a Virtualization Host object to which the API user has object-access permission. If the Virtualization Host is of type **"zvm"** and its *status* is **"not-activated"**, a 409 (Conflict) status code is returned.

If the Request Body Contents fail to validate, a 400 (Bad Request) status code is returned. This may occur because the document defines a field that is not supported for the given virtual server type or includes a field that is not supported because a prerequisite is not met (e.g. attempting to set **maximum-share-mode** when **share-limit** is **"none"**).

If the Request Body Contents are valid, the virtual server's properties are updated to their corresponding request body content's field's values. All fields are optional and may be excluded from the request body; if an optional field is not found in the request body, its property's value will not be modified. As indicated, a property's value is set to its default value if the field is not included in the request body and a prerequisite or other linked field is changed (e.g. if **initial-share-mode** is changed from **"relative"** to **"absolute"**, and **initial-shares** is not defined in the request body, initial-shares will be defaulted to 10.0).

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server object designated by *{virtual-server-id}*
- Action/task permission to the **Virtual Server Details** task.
- Action/task permission to the **Virtual Server Performance Details** task if the **gpmp-support-enabled** or **cpu-perf-mgmt-enabled** field is to be updated, otherwise action/task permission to the **Virtual Server Details** task for all other fields.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
	101	A virtual server with the name specified in the request body already exists on the Virtualization Host with object-id *{virt-host-id}*.
	103	A "shares" (**initial-shares** or **maximum-shares**) property value type is not correct based on the value of its corresponding "share-mode" (**initial-share-mode** or **maximum-share-mode**). For example, the **initial-share-mode** is defined as **"relative"** but the **initial-shares** value is not an integer.
403 (Forbidden)	1	API user does not have action permission to the **Virtual Server Details** task.
	100	The ensemble's **management-enablement-level** property value does not allow the updating of a provided property.
404 (Not Found)	1	A virtual-server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
409 (Conflict)	1	Virtual server status is not valid to perform the operation (does not allow the updating of a specified virtual server property).
	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/7ba96572-0d72-11e1-892f-f0def14b63af HTTP/1.1
x-api-session: 64zdknfzh24fqw8f9v5g99h96rhrxhodmawtknx4iutmgqrfvf
content-type: application/json
content-length: 169
{
   "auto-start": true,
   "description": "Spacely Sprockets Web Store Web Server #1A",
   "initial-virtual-processors": 2,
   "keyboard-language": "en_US",
   "name": "SS-Web-Svr-1A"
}
```

Figure 107. Update Virtual Server Properties: Request for a virtual server of type "x-hyp"

```
204 No Content
date: Sat, 12 Nov 2011 21:08:38 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 108. Update Virtual Server Properties: Response for a virtual server of type "x-hyp"

Create Network Adapter

The **Create Network Adapter** operation creates a virtual network adapter for the PowerVM, x Hyp, or z/VM virtual server with the given identifier. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/network-adapters

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

Request body contents vary based on virtual server type.

For **"power-vm"**:

Field name	Type	Rqd/Opt	Description
network-uri	String/ URI	Optional	The **network-uri** property of the network-adapter-power object. Default: **null**

For **"x-hyp"**:

Field name	Type	Rqd/Opt	Description
network-uri	String/ URI	Optional	The **network-uri** property of the network-adapter-x-hyp object. Default: **null**
emulation-mode	String Enum	Required	The **emulation-mode** property of the network-adapter-x-hyp object.

For **"zvm"**:

Field name	Type	Rqd/Opt	Description
`virtual-device-address`	String	Required	The **virtual-device-address** property of network-adapter-zvm object
`type`	String	Required	The **type** property of network-adapter-zvm object. The value may not be **"rmc"**.
`interface-type`	String Enum	Required	The **interface-type** property of network-adapter-zvm object
`real-device-address`	String	Optional	The **real-device-address** property of network-adapter-zvm object
`switch-uri`	String/ URI	Required if **interface-type** is any value except **none**	If the **interface-type** is virtual-iedn, or virtual-qdio, specify the **switch-uri** property of the of the network-adapter-zvm object. If the **interface-type** is physical-iedn, physical-qdio, or physical-iqdn, specify the **element-uri** of a network-adapter-prsm object in the list of network adapters from the **network-adapters** property of the z/VM virtual server's **parent** property's virtual server object.
`port-mode`	String Enum	Required if **interface-type** is virtual-iedn or virtual-qdio	The **port-mode** property of network-adapter-zvm object
`vlan-ids`	String	Required if **interface-type** is virtual-qdio	The **vlan-ids** property of network-adapter-zvm object
`network-uris`	Array of String/ URI	Required if **interface-type** is virtual-iedn or physical-iedn	The **network-uris** property of network-adapter-zvm object

Response body contents

On successful completion, the response body contains the URI of the created network-adapter object.

Field name	Type	Description
`element-uri`	String/URI	The **element-uri** property of the created network adapter object.

Description

This operation creates the network adapters for the identified virtual server and then returns the URI of the created object. The response also includes a **Location** header that provides this URI.

If the virtual server is of type **"power-vm"** or **"x-hyp"** and its **status** is neither **"not-operating"**nor **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server object designated by *{virtual-server-id}*
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in “Response body contents.”

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	109	The switch-uri conflicts with the network adapter type
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters HTTP/1.1
x-api-session: 3vkw2ncjqhmmpzo6bsyjojjlfgzd0kjoqbvhskz2m6z7esmjrr
content-type: application/json
content-length: 104
{
   "emulation-mode": "e1000",
   "network-uri": "/api/virtual-networks/f920171e-03f2-11e1-8e8e-0010184c8334"
}
```

Figure 109. Create Network Adapter: Request for a virtual server of type "x-hyp"

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters/
  4a5073e6-0f9b-11e1-94dc-f0def14b63af
cache-control: no-cache
date: Tue, 15 Nov 2011 15:05:48 GMT
content-type: application/json;charset=UTF-8
content-length: 129
{
   "element-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
    network-adapters/4a5073e6-0f9b-11e1-94dc-f0def14b63af"
}
```

Figure 110. Create Network Adapter: Response for a virtual server of type "x-hyp"

Update Network Adapter

The **Update Network Adapter** operation modifies an existing virtual network adapter specified by an object identifier for the PowerVM, x Hyp, z/VM, or PR/SM virtual server with the given identifier.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/network-adapters/{*network-adapter-id*}

URI variables

Variable	Description
{virtual-server-id}	Object ID of the Virtual Server object
{network-adapter-id}	Element ID of the network adapter

Request body contents

Request body contents vary based on virtual server type.

For **"prsm"**:

Field name	Type	Rqd/Opt	Description
`network-uris`	Array of String/ URI	Optional	The **network-uris** property of the network-adapter-prsm object.

For **"power-vm"**:

Field name	Type	Rqd/Opt	Description
`network-uri`	String/ URI	Optional	The **network-uri** property of the network-adapter-power object. Specify a value of **null** to indicate that the network adapter should not be connected to a virtual network.

For **"x-hyp"**:

Field name	Type	Rqd/Opt	Description
network-uri	String/ URI	Optional	The **network-uri** property of the network-adapter-x-hyp object. Specify a value of **null** to indicate that the network adapter should not be connected to a virtual network.
emulation-mode	String Enum	Optional	The **emulation-mode** property of the network-adapter-x-hyp object.

For **"zvm"**:

Field name	Type	Rqd/Opt	Description
virtual-device-address	String	Optional	The **virtual-device-address** property of network-adapter-zvm object
interface-type	String Enum	Optional	The **interface-type** property of network-adapter-zvm object
real-device-address	String	Optional	The **real-device-address** property of network-adapter-zvm object
switch-uri	String/ URI	Optional; Allowed only if **interface-type** is virtual-iedn or virtual-qdio	The **switch-uri** property of network-adapter-zvm object.
port-mode	String Enum	Optional; Allowed only if **interface-type** is virtual-iedn or virtual-qdio	The **port-mode** property of network-adapter-zvm object
vlan-ids	String	Optional; Allowed only if **interface-type** is virtual-qdio	The **vlan-ids** property of network-adapter-zvm object
network-uris	Array of String/ URI	Optional; Allowed only if **interface-type** is virtual-iedn or physical-iedn	The **network-uris** property of network-adapter-zvm object

Description

This operation modifies the network adapters for the identified virtual server and then returns its properties.

If virtual server is of type **"zvm"** and the target network adapter is the Remote Monitoring and Control network adapter (type **"rmc"**), a 400 (Bad Request) is returned.

If the virtual server is of type **"power-vm"** or **"x-hyp"** and its **status** is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned. If the virtual server is of type **"zvm"**, the network adapter **interface-type** is **"virtual-iedn"** or **"virtual-qdio"**, and the virtual-switch to which the network adapter was connected no longer exists, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server.
- Object access permission to the Virtual Network object(s)
- Action/task permission to the **Virtual Server Details** task

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	106	The targeted network adapter is the Resource Monitoring and Control network adapter.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	108	The virtual switch to which the network adapter was connected is no longer defined in z/VM.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters/
  4a5073e6-0f9b-11e1-94dc-f0def14b63af HTTP/1.1
x-api-session: 3vkw2ncjqhmmpzo6bsyjojjlfgzd0kjoqbvhskz2m6z7esmjrr
content-type: application/json
content-length: 105
{
   "emulation-mode": "virtio",
   "network-uri": "/api/virtual-networks/f3aece54-0db8-11e1-9e2c-00215e69dea0"
}
```

Figure 111. Update Network Adapter: Request for a virtual server of type "x-hyp"

```
204 No Content
date: Tue, 15 Nov 2011 15:05:48 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 112. Update Network Adapter: Response for a virtual server of type "x-hyp"

Delete Network Adapter

The **Delete Network Adapter** operation deletes an existing virtual network adapter specified by an object identifier for the PowerVM, x Hyp, or z/VM virtual server with the given identifier. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

DELETE /api/virtual-servers/{*virtual-server-id*}/network-adapters/{*network-adapter-id*}

URI variables

Variable	Description
{virtual-server-id}	Object ID of the Virtual Server object
{network-adapter-id}	Element ID of the network adapter

Description

This operation deletes the network adapters for the identified virtual server.

If virtual server is of type **"zvm"** and the target network adapter is the Remote Monitoring and Control network adapter (type **"rmc"**), a 400 (Bad Request) is returned.

If the virtual server is of type **"power-vm"** or **"x-hyp"** and its status is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server
- Action/task role permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	106	The targeted network adapter is the Resource Monitoring and Control network adapter.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters/
  4a5073e6-0f9b-11e1-94dc-f0def14b63af HTTP/1.1
x-api-session: 3vkw2ncjqhmmpzo6bsyjojjlfgzd0kjoqbvhskz2m6z7esmjrr
```

Figure 113. Delete Network Adapter: Request

```
204 No Content
date: Tue, 15 Nov 2011 15:05:50 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 114. Delete Network Adapter: Response

Reorder Network Adapter

The **Reorder Network Adapter** operation reorders the virtual network adapter defined for the PowerVM or x Hyp virtual server with the given identifier. This operation is not supported for z/VM and PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/reorder-network-adapters

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`network-adapter-uris`	Array of String/ URI	Required	Ordered list of element-uris for the network adapter object. The order of this array is significant.

Description

This operation reorders the network adapters for the identified virtual server.

If the virtual server is of **type "power-vm"** or **"x-hyp"** and its **status** is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server
- Object access permission to the Virtual Network object(s)
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/reorder-network-adapters HTTP/1.1
x-api-session: 3vkw2ncjqhmmpzo6bsyjojjlfgzd0kjoqbvhskz2m6z7esmjrr
content-type: application/json
content-length: 256
{
   "network-adapter-uris": [
      "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters/
      4a5073e6-0f9b-11e1-94dc-f0def14b63af",
      "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/network-adapters/
      582b5d0e-0db7-11e1-b1f1-f0def14b63af"
   ]
}
```

Figure 115. Reorder Network Adapter: Request

```
204 No Content
date: Tue, 15 Nov 2011 15:05:48 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 116. Reorder Network Adapter: Response

Create Virtual Disk

The **Create Virtual Disk** operation adds a virtual disk to the specified virtual server. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/virtual-disks

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server that will own the new virtual disk.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
name	String	Required	The **name** property of the virtual disk object.
description	String	Optional	The **description** property of the virtual disk object.
size	Long	Required	The **size** property of the virtual disk object. Required and only valid for virtual disks with a **type** of **"storage-group-based"**.
type	String Enum	Required	The **type** property of the virtual disk object.
emulation-mode	String Enum	Required	The **emulation-mode** property of the virtual disk object. Required and only valid for all virtual disks for a virtual server with **type** of **x-"hyp"**.
access-mode	String Enum	Required	The **access-mode** property of the virtual disk object. Required and only valid for all virtual disks for a virtual server with **type** of **"zvm"**. Value may not be **"unsupported"**.

Field name	Type	Rqd/Opt	Description
device-address	String (1-4)	Required	The **device-address** property of the virtual disk object. Required for all virtual disks for a virtual server with **type** of **"zvm"**.
read-password	String (0-8)	Optional	The **read-password** property of the virtual disk. For all virtual disks for a virtual server with **type "zvm"** and a virtual disk **type** of either **"fullpack"** or **"storage-group-based"**.
write-password	String (0-8)	Optional	The **write-password** property of the virtual disk. For all virtual disks for a virtual server with **type "zvm"** and a virtual disk **type** of either **"fullpack"** or **"storage-group-based"**.
multi-password	String (0-8)	Optional	The **multiple-write** password property of the virtual disk. For all virtual disks for a virtual server with **type "zvm"** and a virtual disk **type** of either **"fullpack"** or **"storage-group-based"**.
password	String (0-8)	Optional	The password for the linked virtual disk. For virtual disks with a **type** of **"linked"**.
backing-virtualization-host-storage-resource	String/ URI	Required	The **backing-virtualization-host-storage-resource** property of the virtual disk. Required and only valid for virtual disks with a **type** of **"fullpack"**.
source-virtual-disk	String/ URI	Required	The **source-virtual-disk** property of the virtual disk. Required and only valid for virtual disks with a **type** of **"linked"**.
backing-storage-group	String/ URI	Required	The **backing-storage-group** property of the virtual disk. Required and only valid for virtual disks with a **type** of **"storage-group-based"**.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
element-uri	String/URI	The **element-uri** of the newly created Virtual Disk object.

Description

The Create Virtual Disk operation adds a virtual disk to the virtual server specified by the *{virtual-server-id}* portion of the request URI.

Upon successful completion, the **element-uri** field of the Response Body and the value of the **Location** response header identify the new virtual disk.

If this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

The URI path must designate an existing virtual server object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action/task permission to the **Virtual Server Details** action as well, otherwise status code 403 (Forbidden) is returned.

If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

If the virtual server is of **type "power-vm"** or **"x-hyp"** and its **status** is not **"operating"** ,**"not-operating"** or **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server object designated by *{virtual-server-id}*
- Object access permission to the storage resources to be used by the virtual disk:
  - If virtual disk **type** is **"fullpack"**, object access permission to the storage resource used by the virtualization host storage resource object designated by **backing-virtualization-host-storage-resource**.
  - If virtual disk **type** is **"storage-group-based"**, object access permission to the storage resource used by every virtualization host storage resource owned by the virtualization host storage group object designated by **backing-storage-group**.
  - If virtual disk **type** is **"linked"**, object access permission to the storage resources used by the virtual disk designated by **source-virtual-disk** and to the virtual server that owns it.
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and both the response body and the **Location** response header contain the URI of the newly created object.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	107	The **backing-virtualization-host-storage-resource** property does not designate an existing virtualization host storage resource object, or the API user does not have object access permission to it.
	108	The **backing-storage-group** property does not designate an existing virtualization host storage group object, or the API user does not have object access permission to it.
	109	The **source-virtual-disk** property does not designate an existing virtual disk object, or the API user does not have object access permission to it.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
content-type: application/json
content-length: 259
{
   "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
    2f7bf364-03f8-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
    37699380-0fa9-11e1-b69e-f0def14b63af",
   "description": "Boot filesystem",
   "name": "boot",
   "type": "fullpack"
}
```

Figure 117. Create Virtual Disk: Request for a virtual server of type "power-vm"

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
  c547fb4e-0fac-11e1-842b-f0def14b63af
cache-control: no-cache
date: Tue, 15 Nov 2011 17:10:57 GMT
content-type: application/json;charset=UTF-8
content-length: 126
{
   "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
    c547fb4e-0fac-11e1-842b-f0def14b63af"
}
```

Figure 118. Create Virtual Disk: Response for a virtual server of type "power-vm"

Delete Virtual Disk

The **Delete Virtual Disk** operation removes a specified virtual disk from the specified virtual server. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

DELETE /api/virtual-servers/{*virtual-server-id*}/virtual-disks/{*virtual-disk-id*}

URI variables

Variable	Description
{virtual-server-id}	Object ID of the virtual server object
{virtual-disk-id}	Element ID of the virtual disk object

Description

The **Delete Virtual Disk** operation removes a specified virtual disk from the specified virtual server. The virtual disk is identified by the *{virtual-disk-id}* variable in the URI, and the virtual server is identified by the *{virtual-server-id}* variable in the URI.

Upon successfully removing the virtual disk, HTTP status code 204 (No Content) is returned and no response body is provided.

The URI path must designate an existing virtual server object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtual disk object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action/task permission to the Virtual Server Details action as well, otherwise status code 403 (Forbidden) is returned.

If the virtual server is of **type "power-vm"** or **"x-hyp"** and its **status** is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server object designated by *{virtual-server-id}*.
- If virtual disk **type** is **"fullpack"**, object access permission to the virtualization host storage resource object designated by **backing-virtualization-host-storage-resource**
- If virtual disk **type** is **"storage-group-based"**, object access permission to all virtualization host storage resources owned by the virtualization host group object designated by **backing-storage-group**.
- If virtual disk **type** is **"linked"**, object access permission to virtual disk designated by **source-virtual-disk** and the virtual server that owns it.
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual server object, or the API user does not have object access permission to it.
	2	The object ID in the URI *{virtual-disk-id}* does not designate an existing virtual disk object. (Bad URI Object ID)
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
  c547fb4e-0fac-11e1-842b-f0def14b63af HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
```

Figure 119. Delete Virtual Disk: Request

```
204 No Content
date: Tue, 15 Nov 2011 17:11:03 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 120. Delete Virtual Disk: Response

Get Virtual Disk Properties

The **Get Virtual Disk Properties** operation retrieves the properties of a single virtual disk object that is designated by its object ID and the object ID of the owning virtual server. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

GET /api/virtual-servers/{*virtual-server-id*}/virtual-disks/{*virtual-disk-id*}

URI variables

Variable	Description
{virtual-server-id}	Object ID of the virtual server that will own the virtual disk.
{virtual-disk-id}	Element ID of the virtual disk object for which properties are to be obtained.

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the virtual disk object as defined in the Data Model section above. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

This operation returns the current properties for the virtual disk object specified by *{virtual-disk-id}*.

On successful execution, all of the current properties as defined by the "Data Model" on page 206 are provided in the response body and HTTP status code 200 (OK) is returned.

The URI path must designate an existing virtual server object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtual disk object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have

action/task permission to the **Virtual Server Details** action, otherwise status code 403 (Forbidden) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server object designated by *{virtual-server-id}*
- If virtual disk **type** is **"fullpack"**, object access permission to the virtualization host storage resource object designated by **backing-virtualization-host-storage-resource**
- If virtual disk **type** is **"storage-group-based"**, object access permission to all virtualization host storage resources owned by the virtualization host storage group object designated by **backing-storage-group**.
- If virtual disk **type** is **"linked"**, object access permission to virtual disk designated by **source-virtual-disk** and the virtual server that owns it.
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual server object, or the API user does not have object access permission to it.
	2	The object ID in the URI *{virtual-disk-id}* does not designate an existing virtual disk object.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
  c547fb4e-0fac-11e1-842b-f0def14b63af HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
```

Figure 121. Get Virtual Disk Properties: Request for a virtual server of type "power-vm"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 17:10:57 GMT
content-type: application/json;charset=UTF-8
content-length: 516
{
   "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/
    2f7bf364-03f8-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
    37699380-0fa9-11e1-b69e-f0def14b63af",
   "description": "Boot filesystem",
   "element-id": "c547fb4e-0fac-11e1-842b-f0def14b63af",
   "element-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
    c547fb4e-0fac-11e1-842b-f0def14b63af",
   "name": "boot",
   "owner": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af",
   "size": 17179869184,
   "type": "fullpack"
}
```

Figure 122. Get Virtual Disk Properties: Response for a virtual server of type "power-vm"

Update Virtual Disk Properties

The **Update Virtual Disk Properties** operation updates one or more of the writeable properties of a virtual disk object. This operation is not supported for PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/virtual-disks/{*virtual-disk-id*}

URI variables

Variable	Description
{virtual-server-id}	Object ID of the virtual server that owns the virtual disk.
{virtual-disk-id}	Element ID of the virtual disk object for which properties are to be updated.

Request body contents

The request body is expected to contain a JSON object that provides the new values of any writeable property that is to be updated by this operation. Field names and data types in this JSON object are expected to match the corresponding property names and data types defined in the data model for this object type. The JSON object can and should omit fields for properties whose values are not to be changed by this operation. All such fields are optional.

The JSON object may also contain the following non-data-model field:

Field name	Type	Rqd/Opt	Description
`password`	String (0-8)	Optional	The new password for a linked virtual disk. For virtual disks with a **type** of **"linked"**

Request body **access-mode** property value may not be **"unsupported"**.

Description

This operation updates writeable properties of the virtual disk object specified by *{virtual-disk-id}*.

The request body contains an object with one or more fields with field names that correspond to the names of properties for this object. On successful execution, the value of each corresponding property of the object is updated with the value provided by the input field, and status code 204 (No Content) is returned without supplying any response body. The request body does not need to specify a value for all writeable properties, but rather can and should contain only fields for the properties to be updated. Object properties for which no input value is provided remain unchanged by this operation.

If the update changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

The URI path must designate an existing virtual disk object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtual server object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action/task permission to the **Virtual Server Details** action as well, otherwise status code 403 (Forbidden) is returned.

The request body is validated against the data model for this object type to ensure that it contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

If the virtual server is of **type "power-vm"** or **"x-hyp"** and its **status** is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server object designated by *{virtual-server-id}*
- If virtual disk **type** is **"fullpack"**, object access permission to the virtualization host storage resource object designated by **backing-virtualization-host-storage-resource**
- If virtual disk **type** is **"storage-group-based"**, object access permission to all virtualization host storage resources owned by the virtualization host storage group object designated by **backing-storage-group**.
- If virtual disk **type** is **"linked"**, object access permission to virtual disk designated by **source-virtual-disk** and the virtual server that owns it.
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	The API user does not have the required permission for this operation.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual server object, or the API user does not have object access permission to it.
	2	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual disk object.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
  c1418fba-0fac-11e1-903e-f0def14b63af HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
content-type: application/json
content-length: 44
{
   "emulation-mode": "virtio",
   "name": "pv01"
}
```

Figure 123. Update Virtual Disk Properties: Request for a virtual server of type "x-hyp"

```
204 No Content
date: Tue, 15 Nov 2011 17:10:50 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 124. Update Virtual Disk Properties: Response for a virtual server of type "x-hyp"

Reorder Virtual Disks

The **Reorder Virtual Disks** operation reorders the virtual disks defined for the PowerVM or x Hyp virtual server with the given identifier. This operation is not supported for z/VM and PR/SM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/reorder-virtual-disks

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

The JSON object may also contain the following field:

Field name	Type	Rqd/Opt	Description
`virtual-disk-uris`	Array of String/ URI	Required	Ordered list of virtual disk object **element-uri** property values. The order of URIs in the array defines the new order of the virtual disks. The array size must equal the number of virtual disks for the virtual server and the array must include the **element-uri** of each virtual disk in the virtual server exactly once.

Description

This operation reorders the virtual disks for the identified virtual server.

If the URI path does not designate an existing virtual server object or the API user does not have object access permission to that virtual server and the storage-resource objects backing its virtual disks, a 404 (Not Found) status code is returned. If the virtual server is of **type "zvm"** or **"prsm"**, a 400 (Bad Request) status code is returned. If the API user does not have action/task permission to the **Virtual Server Details** action, a 403 (Forbidden) status code is returned.

If **virtual-disk-uris** array size is not equal to the number of virtual disks in the virtual server or the array does not include the element-uri of each virtual disk in the virtual server, a 400 (Bad Request) status code is returned.

If the virtual server is of **type "power-vm"** or **"x-hyp"** and its status is neither **"not-operating"** nor **"not-activated"**, a 409 (Conflict) status code is returned.

If the request body contents are valid, the virtual server's virtual disks are reordered to match the order of their **element-uri** properties in the **virtual-disk-uris** array.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server object designated by *{virtual-server-id}*
- Object access permission to the storage resources used by each virtual disk in the virtual server:
  - If virtual disk **type** is **"fullpack"**, object access permission to the storage resource used by the virtualization host storage resource object designated by **backing-virtualization-host-storage-resource**.
  - If virtual disk **type** is **"storage-group-based"**, object access permission to the storage resource used by every virtualization host storage resource owned by the virtualization host storage group object designated by **backing-storage-group**.
  - If virtual disk **type** is **"linked"**, object access permission to the storage resources used by the virtual disk designated by **source-virtual-disk** and the virtual server that owns it.
- Action/task permission to the **Virtual Server Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
	106	The **virtual-disk-uris** array contains an invalid number values.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual server object, or the API user does not have object access permission to it.
	2	The object ID in the URI *{virtual-server-id}* does not designate an existing virtual disk object.
409 (Conflict)	1	Virtual server status is not valid to perform the operation.
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM Virtualization Host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM Virtualization Host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/reorder-virtual-disks HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
content-type: application/json
content-length: 247
{
   "virtual-disk-uris": [
      "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
      c1418fba-0fac-11e1-903e-f0def14b63af",
      "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/virtual-disks/
      c1149ce4-0fac-11e1-903e-f0def14b63af"
   ]
}
```

Figure 125. Reorder Virtual Disks: Request

```
204 No Content
date: Tue, 15 Nov 2011 17:10:50 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 126. Reorder Virtual Disks: Response

Activate Virtual Server

The **Activate Virtual Server** operation accepts a request to asynchronously activate the identified virtual server.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/activate

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Response body contents

Once the activation request is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve activation status updates.

Asynchronous result description

Once the activation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Activate Virtual Server** request.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in the operation description. The **job-results** field is null for asynchronous virtual server activation jobs.

Description

This operation asynchronously activates the identified virtual server. If the URI does not identify a valid virtual server a 404 (Not Found) status code is returned. If the user does not have authority to perform the Activate action, a 403 (Forbidden) status code is returned.

The virtual server activation job is then initiated and a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the activation job. See “Query Job Status” on page 44 for information on how to query job status. When the activate job has completed, an asynchronous result message is sent, including “Job status and reason codes” on page 278.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server
- Action/task permission to the **Activate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in “Response body contents.”

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have access action permission to **Activate**.
404 (Not Found)	1	A virtual server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
200 (OK)	N/A	Activation completed successfully
500 (Server Error)	100	Virtual server activation failed
	101	Virtual server activation job timed out

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/activate HTTP/1.1
x-api-session: 1f6hv807yexwhfyk1p8ygrc8876y48adda2dfpvuz4t9iqeo9k
```

Figure 127. Activate Virtual Server: Request

```
202 Accepted
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 16 Nov 2011 18:49:07 GMT
content-type: application/json;charset=UTF-8
content-length: 60
{
   "job-uri": "/api/jobs/aa24c23e-1083-11e1-87b6-0010184c8334"
}
```

Figure 128. Activate Virtual Server: Response

Deactivate Virtual Server

The **Deactivate Virtual Server** operation accepts a request to asynchronously deactivate the identified virtual server.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/deactivate

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Response body contents

Once the deactivation request is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve deactivation status updates.

Asynchronous result description

Once the deactivation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Deactivate Virtual Server** request.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description below. The **job-results** field is null for asynchronous virtual server deactivation jobs.

Description

This operation asynchronously deactivates the identified virtual server. If the URI does not identify a valid virtual server a 404 (Not Found) status code is returned. If the user does not have authority to perform the Deactivate action, a 403 (Forbidden) status code is returned.

The virtual server deactivation job is then initiated and a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the deactivation job. See "Query Job Status" on page 44 for information on how to query job status. When the deactivate job has completed, an asynchronous result message is sent, including "Job status and reason codes" on page 280.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server
- Action/task permission to the **Deactivate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have access action permission to **Activate**.
404 (Not Found)	1	A virtual server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
200 (OK)	N/A	Deactivation completed successfully
500 (Server Error)	100	Virtual server deactivation failed
	101	Virtual server deactivation job timed out

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/deactivate HTTP/1.1
x-api-session: 1f6hv807yexwhfyk1p8ygrc8876y48adda2dfpvuz4t9iqeo9k
```

Figure 129. Deactivate Virtual Server: Request

```
202 Accepted
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 16 Nov 2011 18:50:49 GMT
content-type: application/json;charset=UTF-8
content-length: 60
{
   "job-uri": "/api/jobs/e6eb3504-1083-11e1-87b6-0010184c8334"
}
```

Figure 130. Deactivate Virtual Server: Response

Mount Virtual Media

The **Mount Virtual Media** operation starts an asynchronous operation to mount the specified zManager-provided ISO to the identified PowerVM or x Hyp virtual server. This operation is not supported for PR/SM and z/VM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/mount-virtual-media

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`iso`	String Enum	The ID of the zManager-provided ISO to mount to the virtual server. Values: • **"gpmp"**– Guest Platform Performance Management ISO • **"virtio"** – RedHat VirtIO driver CD

Asynchronous result description

Once the mount job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Mount Virtual Media** request.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description below. The **job-results** field is null for asynchronous mount virtual media jobs.

Description

The Mount Virtual Media operation starts an asynchronous operation to mount the specified zManager-provided ISO to the identified PowerVM or x Hyp virtual server. This operation is not supported for PR/SM and z/VM virtual servers.

Virtual media is a virtual DVD drive that can be attached to a virtual server. Because virtual media is allocated from IBM blade storage instead of on a real media, it may be faster to use than a physical real device.

Preloaded ISO images come in two variants. The first is an ISO image that is packaged on the SE, such as the GPMP Installation Image. When mounting this type of ISO image to the virtual server, the image will be uploaded from the SE to the hypervisor. The second type is an ISO image preloaded onto the hypervisor firmware, such as the Red Hat VirtIO Driver Image. When mounting this type of ISO image, no uploading will occur.

The response from this operation, success or failure, will be presented asynchronously. Once the mount operation job has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body from the **Mount Virtual Media** request.

If the API user does not have action permission for the **Mount Virtual Media** task, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virtual-server-id}* does not identify a virtual server object to which the API user has object-access permission.

A mount operation cannot be performed when a virtual server or its Virtualization Host is busied by another operation or when the Virtualization Host **status** is not **"operating"**. Under these conditions a 409 (Conflict) status code is returned.

The mount virtual media job is then initiated and a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the mount job. See “Query Job Status” on page 44 for information on how to query job status. When the mount job has completed, an asynchronous result message is sent, including “Job status and reason codes” on page 282.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server
- Action/task permission to the **Mount Virtual Media** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	API user does not have access action permission to **Mount Virtual Media**.
404 (Not Found)	1	A virtual server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
409 (Conflict)	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	101	Parent Virtualization Host has a status value that is not valid to perform the operation.
	105	Parent Virtualization Host object was locked/busy and the request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
200 (OK)	N/A	Mount completed successfully
500 (Server Error)	100	Mount failed

Usage notes

- The mount operation may take some time as the zManager-provided ISO may need to be internally uploaded to the hosting virtualization host from other elements in the management hierarchy.
- Upload and mount of user-provided ISOs is supported by the **Mount Virtual Media Image** operation available in version 1.2 of the Web Services API.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/mount-virtual-media HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
content-type: application/json
content-length: 15
{
   "iso": "gpmp"
}
```

Figure 131. Mount Virtual Media: Request

```
202 Accepted
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 15 Nov 2011 17:10:50 GMT
content-type: application/json;charset=UTF-8
content-length: 60
{
   "job-uri": "/api/jobs/c4e59922-0fac-11e1-bfcd-00215e69dea0"
}
```

Figure 132. Mount Virtual Media: Response

Mount Virtual Media Image

The **Mount Virtual Media Image** operation starts a synchronous operation to mount a user-provided ISO image to the identified PowerVM or x Hyp virtual server. This operation is not supported for PR/SM and z/VM virtual servers. Virtual media is a virtual DVD drive that can be attached to a virtual server. Because virtual media is allocated from IBM blade storage instead of on a real media, it may be faster to use than a physical real device. The contents of the ISO image is specified as binary data in the body of the POST request.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/mount-virtual-media-image

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Query parameters:

Name	Type	Rqd/Opt	Description
image-name	String	Optional	If specified, this will be used as the displayable name and returned as the mounted-media-name in the virtual server properties. If omitted, the service will generate a name for this use.

Request body contents

The request body is the binary contents of an ISO image file. A MIME media type of **application/octet-stream** should be specified as the content-type on the request.

Description

This operation mounts the provided ISO image content to the identified virtual server.

If the API user does not have action permission for the **Mount Virtual Media** task, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virtual-server-id}* does not identify a virtual server object to which the API user has object-access permission.

A mount operation cannot be performed when a virtual server or its virtualization host is busied by another operation or when the virtualization host status is not **"operating"**. Under these conditions a 409 (Conflict) status code is returned.

If the mount completes successfully a 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the virtual server.
- Action/task permission to the **Mount Virtual Media** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	API user does not have action permission to the **Mount Virtual Media** task.
404 (Not Found)	1	A virtual server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
409 (Conflict)	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	101	Parent Virtualization Host has a status value that is not valid to perform the operation.
	105	Parent Virtualization Host object was locked/busy and the request timed out.
500 (Server Error)	100	Mount failed.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- Because the ISO image is transmitted synchronously as part of the request body of this operation, the time to execute this operation can be expected to be, at least in part, proportional to the size of the ISO image being uploaded.
- Mount of preloaded ISO images is supported by the **Mount Virtual Media** operation of the Web Services API.

Unmount Virtual Media

The **Unmount Virtual Media** operation unmounts the currently mounted ISO from the identified PowerVM or x Hyp virtual server. This operation is not supported for PR/SM and z/VM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/unmount-virtual-media

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

An optional request body can be specified as a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	If true, a forced unmount is requested. Otherwise, a normal unmount is requested. Default is false.

Description

This operation unmounts the currently mounted ISO from the identified virtual server. Note that there are not separate unmount operations associated with the distinct **Mount Virtual Media** and **Mount Virtual Media Image** operations; this single **Unmount Virtual Media** operation may be used regardless of the mount operation used.

If the API user does not have action/task permission for the **Mount Virtual Media** task, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virtual-server-id}* does not identify a virtual server object to which the API user has object-access permission.

If a request body is provided with the **force** request field specified as true, a request is issued to the virtualization host to attempt to force the unmount even if the media is locked by a guest OS. Otherwise, a normal unmount operation is performed.

An unmount operation cannot be performed when a virtual server or its virtualization host is busied by another operation or when the virtualization host **status** is not **"operating"**. Under these conditions a 409 (Conflict) status code is returned.

The virtual media is then unmounted and a 204 (No Content) status code is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server.
- Action/task permission to the **Mount Virtual Media** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	API user does not have action permission to the **Mount Virtual Media** task.
404 (Not Found)	1	A virtual server with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
409 (Conflict)	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	101	Parent Virtualization Host has a status value that is not valid to perform the operation.
	105	Parent Virtualization Host object was locked/busy and the request timed out.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/operations/unmount-virtual-media HTTP/1.1
x-api-session: 29chofb3vqxpijctnd2kf5fiwnfll8z62mq3yomtqjyz8bx2gn
```

Figure 133. Unmount Virtual Media: Request

```
204 No Content
date: Tue, 15 Nov 2011 17:10:55 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 134. Unmount Virtual Media: Response

Migrate Virtual Server

The **Migrate Virtual Server** operation moves the identified PowerVM or x Hyp virtual server to the target virtualization host. This operation is not supported for PR/SM and z/VM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/migrate

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`virt-host-id`	String	Object ID of the target virtualization host object.

Response body contents

On successful completion, a response body is provided such that the caller may further target the migrated virtual server.

Field name	Type	Description
`object-uri`	String/URI	The **object-uri** property of the migrated Virtual Server object on the new virtualization host.

Description

This operation moves the identified virtual server to a new virtualization host.

Virtual server migration is a composite action consisting of the steps of creating a new virtual server on the new virtualization host, configuring the new virtual server to correspond to the identified (original) virtual server, and then deleting the identified virtual server. Upon completion, this operation returns the **object-uri** of the new virtual server in the response body. The response also includes a **Location** header that provides this URI.

The steps of the composite migration action may be observable through inventory and property change notifications.

If the API user does not have action permission for the Migrate Virtual Server task, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virtual-server-id}* does not identify a virtual server object to which the API user has object-access permission. A 400 (Bad Request) status code is returned if the request body fails to validate (e.g. the target virtualization host is of a different type than the current virtualization host).

A migrate operation cannot be performed when the virtual server, its storage-resources, or virtual-networks are busied by another operation. Migrate is also not possible when the virtual server's **status** is something other than **"not-operating"** or **"not-activated"** or when its current virtualization host or the target virtualization host **status** is **"not-communicating"** or **"status-check"**. Migration is also not possible when the virtual server has media mounted. Under any of these conditions a 409 (Conflict) status code is returned.

Virtual server migration is then attempted. Migration may fail if the user does not have object-access to the storage resources used by the virtual server's virtual disks. Migration may also fail if the virtual server has virtual disks and the target virtualization host does not have virtualization host storage resources for them. If these or other failure occurs, migration fails, the original virtual server is restored, and a 503 (Service Unavailable) status code is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server
- Object access permission to the virtual server's virtualization host's hosting-environment
- Object access permission to the target virtualization host's hosting-environment

- Object access permission to the storage resources used by the virtual server's virtual disks
- Action/task permission to the **Migrate Virtual Server** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents" on page 287.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
	104	Target virtualization host is invalid because it is not the same type as the current virtualization host.
403 (Forbidden)	1	API user does not have action permission to the **Migrate Virtual Server** task.
404 (Not Found)	1	A virtual server object with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it or its virtualization host's hosting-environment.
	1	A virtualization host with object-id *{virt-host-id}* does not exist on HMC or API user does not have object-access permission for its hosting-environment.
409 (Conflict)	0	Migration is not possible for a reason other than those indicated by the other 409 (Conflict) reason codes.
	1	Virtual server status is not valid to perform the operation (must be either **"not-operating"** or **"not-activated"**).
	2	Virtual server object with ID *{virtual-server-id}* was busy and request timed out.
	101	Parent Virtualization Host has a status value that is not valid to perform the operation.
	103	Virtual server is an invalid target because it has virtual media is mounted.
	105	The target virtualization host already has a virtual server with the same name as the virtual server to be migrated.
	106	One or more of the virtual server's storage resources was busy and the request timed out.
	107	One or more of the virtual server's virtual network objects was busy and the request timed out.
	108	One or more storage resources used by the virtual server are not available on the target virtualization host.
503 (Service Unavailable)	0	Migration failed.
	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Initiate Virtual Server Dump

The **Initiate Virtual Server Dump** operation disruptively stops the identified PowerVM or x Hyp virtual server and sends an operating system-specific command to begin a memory dump of the operating system running on the virtual server. This operation is not supported for PR/SM or z/VM virtual servers.

HTTP method and URI

POST /api/virtual-servers/{*virtual-server-id*}/operations/initiate-dump

In this request, the URI variable *{virtual-server-id}* is the object ID of the virtual server object.

Description

The Initiate Virtual Server Dump operation disruptively stops the virtual server and initiates a dump.

Virtual server **status** must not be **"not-operating"**.

This operation disruptively stops the identified virtual server and sends an virtual-server-type-specific signal to begin a memory dump of the operating system running on the virtual server.

If the API user does not have action permission for the Initiate Virtual Server Dump task, a 403 (Forbidden) status code is returned. A 404 (Not Found) status code is returned if the object-id *{virtual-server-id}* does not identify a virtual server object to which the API user has object-access permission. A 400 (Bad Request) status code is returned if the request body fails to validate or if the operation is not supported for the given type of virtual server.

A 409 (Conflict) status code is returned if the virtual server status is **"not-operating"**.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the virtual server
- Action/task permission to the **Initiate Virtual Server Dump** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	100	The operation does not support a virtual server of the given type.
403 (Forbidden)	1	API user does not have action permission to the **Initiate Virtual Server Dump** task.
404 (Not Found)	1	A virtual server object with object-id *{virtual-server-id}* does not exist on HMC or API user does not have object-access permission for it.
409 (Conflict)	1	Virtual server status is not valid to perform the operation (status is **"not-operating"**).

HTTP error status code	Reason code	Description
	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/operations/initiate-dump HTTP/1.1
x-api-session: 6cq5qn64f6zv388z20hu6nzpx1ukaqrl4ekri6sgezaxcwgtz0
```

Figure 135. Initiate Virtual Server Dump: Request

```
204 No Content
date: Wed, 07 Dec 2011 05:07:39 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 136. Initiate Virtual Server Dump: Response

Inventory service data

Information about the virtual servers managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for Virtual Server objects are included in the response to the Inventory Service's Get Inventory operation when the request specifies (explicitly by inventory class, implicitly via a containing category, or by default) that objects of the various virtual server type-specific inventory classes are to be included. An entry for a particular Virtual Server is included only if the API user has object-access permission to that object and the applicable type-specific inventory class has been specified, as described in the following table:

Inventory class	Includes virtual servers with “type” value
power-vm-virtual-server power-vm-virtual-server-common	power-vm
prsm-virtual-server prsm-virtual-server-common	prsm
x-hyp-virtual-server x-hyp-virtual-server-common	x-hyp
zvm-virtual-server zvm-virtual-server-common	zvm

For each Virtual Server object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for the **Get Virtual**

Server Properties operation. That is, the data provided is the same as would be provided if a Get Virtual Server Properties operation were requested targeting this object. For inventory class names that end with **–common** (e.g. **power-vm-virtual-server-common**), the data is the same as would be provided for a Get Virtual Sever Properties operation with the **properties=common** query parameter specified. For the inventory class names that do not end in **–common** (e.g. **power-vm-virtual-server**), the results are the same as would be provided for a Get Virtual Server Properties operation with no **properties** query parameter specified.

Sample inventory data

```
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start": false,
   "boot-mode": "normal",
   "boot-network-adapter-client-ip": null,
   "boot-network-adapter-gateway-ip": null,
   "boot-network-adapter-server-ip": null,
   "boot-network-adapter-subnet-ip": null,
   "boot-sequence": [
      "virtual-disk"
   ],
   "class": "virtual-server",
   "cpu-perf-mgmt-enabled": true,
   "description": "Order processing for the Shimmer floor wax/dessert topping product",
   "dlpar-active": false,
   "dlpar-enabled": false,
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": true,
   "hostname": null,
   "initial-dedicated-processors": null,
   "initial-memory": 1024,
   "initial-processing-units": 0.10000000000000001,
   "initial-virtual-processors": 1,
   "is-locked": false,
   "keylock": "normal",
   "mac-prefix": {
      "mac-address": "02:ee:6b:6c:70:00",
      "prefix-length": 40
   },
   "maximum-dedicated-processors": null,
   "maximum-memory": 1024,
   "maximum-processing-units": 7.0,
   "maximum-virtual-processors": 7,
   "minimum-dedicated-processors": null,
   "minimum-memory": 1024,
   "minimum-processing-units": 0.10000000000000001,
   "minimum-virtual-processors": 1,
   "mounted-media-name": null,
   "name": "Shimmer orders server",
   "network-adapters": [
      {
```

Figure 137. Virtual server object: Sample inventory data for a virtual server of type "power-vm" (Part 1)

```
            "element-id": "0000",
            "element-uri": "/api/virtual-servers/42eecf00-7b47-11e0-bc9e-001f163803de/network-adapters/0000",
            "network-uri": "/api/virtual-networks/9b9fe4f8-75b2-11e0-9219-0010184c8026"
         },
         {
            "element-id": "0010",
            "element-uri": "/api/virtual-servers/42eecf00-7b47-11e0-bc9e-001f163803de/network-adapters/0010",
            "network-uri": null
         }
      ],
      "object-id": "119338a2-4081-11e0-9e7c-f0def10bff8d",
      "object-uri": "/api/virtual-servers/119338a2-4081-11e0-9e7c-f0def10bff8d",
      "parent": "/api/virtualization-hosts/baab1cd2-2990-11e0-8d5b-001f163803de",
      "processing-mode": "shared",
      "status": "not-operating",
      "type": "power-vm",
      "virtual-disks": [
         {
            "backing-virtualization-host-storage-resource": "/api/virtualization-hosts/baab1cd2-2990-11e0-
         8d5b-001f163803de/virtualization-host-storage-resources/abfce790-4080-11e0-8486-f0def10bff8d",
            "description": "",
            "element-id": "11b5b0a8-4081-11e0-8486-f0def10bff8d",
            "element-uri": "/api/virtual-servers/119338a2-4081-11e0-9e7c-f0def10bff8d/virtual-disks/
         11b5b0a8-4081-11e0-8486-f0def10bff8d",
            "name": "Superman5",
            "owner": "/api/virtual-servers/119338a2-4081-11e0-9e7c-f0def10bff8d",
            "size": 5242880,
            "type": "fullpack"
         }
      ],
      "workloads": [
         "/api/workload-resource-groups/a4019e06-7685-11e0-8fca-0010184c8026"
      ]
   }
```

Figure 138. Virtual server object: Sample inventory data for a virtual server of type "power-vm" (Part 2)

```
    "acceptable-status": [
       "operating"
    ],
    "associated-logical-partition": "/api/logical-partitions/0b239aa3-fea1-32e0-a38f-c632e7ee3b0c",
    "class": "virtual-server",
    "cpu-perf-mgmt-enabled": false,
    "description": "",
    "gpmp-status": "not-operating",
    "gpmp-support-enabled": false,
    "gpmp-version": "unavailable",
    "has-unacceptable-status": true,
    "is-locked": false,
    "name": "VMALT2  ",
    "network-adapters": [
       {
          "chpid": "F0",
          "css": "0",
          "element-id": "OSX 0.F0",
          "element-uri": "/api/virtual-servers/d44575cc-40ea-11e0-9814-001f163803de/network-adapters/
       OSX%200.F0",
          "name": "OSX 0.F0",
          "type": "osx"
       },
       {
          "chpid": "A0",
          "css": "0",
          "element-id": "OSX 0.A0",
          "element-uri": "/api/virtual-servers/d44575cc-40ea-11e0-9814-001f163803de/network-adapters/
       OSX%200.A0",
          "name": "OSX 0.A0",
          "network-uris": [
             "/api/virtual-networks/a9b6f8ce-771f-11e0-b1da-0010184c8026"
          ],
          "type": "osx"
       }
    ],
    "object-id": "d44575cc-40ea-11e0-9814-001f163803de",
    "object-uri": "/api/virtual-servers/d44575cc-40ea-11e0-9814-001f163803de",
    "parent": "/api/virtualization-hosts/bab76208-2990-11e0-8d5b-001f163803de",
    "status": "operating",
    "type": "prsm",
    "workloads": [
       "/api/workload-resource-groups/a4019e06-7685-11e0-8fca-0010184c8026"
    ]
}
```

Figure 139. Virtual server object: Sample inventory data for a virtual server of type "prsm"

```
{
   "acceptable-status": [
      "operating"
   ],
   "auto-start": false,
   "boot-sequence": [
      "virtual-media"
   ],
   "class": "virtual-server",
   "description": "",
   "gpmp-status": "not-operating",
   "gpmp-support-enabled": false,
   "gpmp-version": "unavailable",
   "has-unacceptable-status": true,
   "hostname": null,
   "initial-memory": 4096,
   "initial-virtual-processors": 4,
   "is-locked": false,
   "mounted-media-name": "ubuntu-11.04-server-i386.iso",
   "name": "XVS1",
   "network-adapters": [],
   "object-id": "a4588932-8648-11e0-bbc1-f0def10bff8d",
   "object-uri": "/api/virtual-servers/a4588932-8648-11e0-bbc1-f0def10bff8d",
   "parent": "/api/virtualization-hosts/931b25d6-82e1-11e0-b9e4-f0def10bff8d",
   "status": "stopping",
   "type": "x-hyp",
   "virtual-disks": [],
   "workloads": [
      "/api/workload-resource-groups/a4019e06-7685-11e0-8fca-0010184c8026"
   ]
}
```


Figure 140. Virtual server object: Sample inventory data for a virtual server of type "x-hyp"

Chapter 11. Storage Management

zManager provides a common interface across the different Virtualization Host types and storage types that it supports. It allows a system administrator to create virtualized storage resources and attach them to virtual servers. The basic flow for all supported types of Virtualization Hosts and storage resources is as follows:

The server administrator defines his requirements to the storage administrator (e.g., the number of storage resources, their type and size information).

The server administrator uses zManager storage management interfaces to export the virtualization-host-specific information which is required to allow the Storage Area Network (SAN) administrator to setup the SAN accordingly. This information consists primarily of the Host WWPN List.

When the SAN administrator has finished configuring storage resources, the server administrator can add these new storage resources to the ensemble for management by performing one or more of the following actions:

- Triggering discovery of newly detected storage resources for a Virtualization Host
- Compiling a Storage Access List (a file with information about the storage resources, such as unique name and addressing information) and importing this list into zManager
- Manually adding each storage resource.

zManager offers interfaces to work with all ensemble-managed storage resources. For example, there are interfaces to:

- List the various storage-related entities (storage resources, Virtualization Host storage resources, Virtualization Host storage groups and virtual disks)
- List details of the various storage-related entities
- Identify storage resources that are to be managed by zManager
- Grant Virtualization Hosts access to storage resources
- Assign storage resources to a Virtualization Host storage group
- Assign storage resources to virtual servers by creating virtual disks on them.

Terms

Host World Wide Port Name (WWPN) List

The Host WWPN List consists of a list of WWPNs of the Fibre-Channel host ports of each virtualization host. It can be exported for one or multiple virtualization hosts through a zManager function. The WWPN list is useful when the system administrator requires additional storage resources to be configured by the storage administrator. The storage administrator must enter these WWPNs into the SAN switches and storage controllers in order to allow these specific WWPNs to access the storage controllers / Logical Units.

Storage Access List (SAL)

The Storage Access List is provided by a storage administrator to a system administrator after configuring storage resources (e.g., FCP Logical Units). The Storage Access List consists of a number of host port WWPNs and entries for a configured storage resource with its properties, such as addressing information, or device type information, in the form of a Comma-Separated Values (CSV) file. Importing a Storage Access List offers a convenient way for letting zManager know which hypervisor has access to which storage resource, avoiding the cumbersome and error-prone process of adding storage resources (and their associated properties) manually.

Storage Resource
An addressable storage entity, allowing a virtualization host to write data to and read data from. A storage resource may be one of the following: a SCSI Logical Unit, attached via FCP, a file, a Volume attached via ESCON/FICON.

Virtualization Host Storage Resource
The representation of a storage resource from the perspective of a virtualization host. It is a storage resource to which the virtualization host has access.

Virtualization Host Storage Group
Representation of a z/VM Storage Group in zManager. It consists of homogeneous storage resources to which a z/VM virtualization host has access.

Virtual Disk
Virtual storage space provided by a virtualization host to a guest virtual server. A Virtual Disk is based upon a storage resource, but may be further virtualized by a virtualization host.

Object model overview

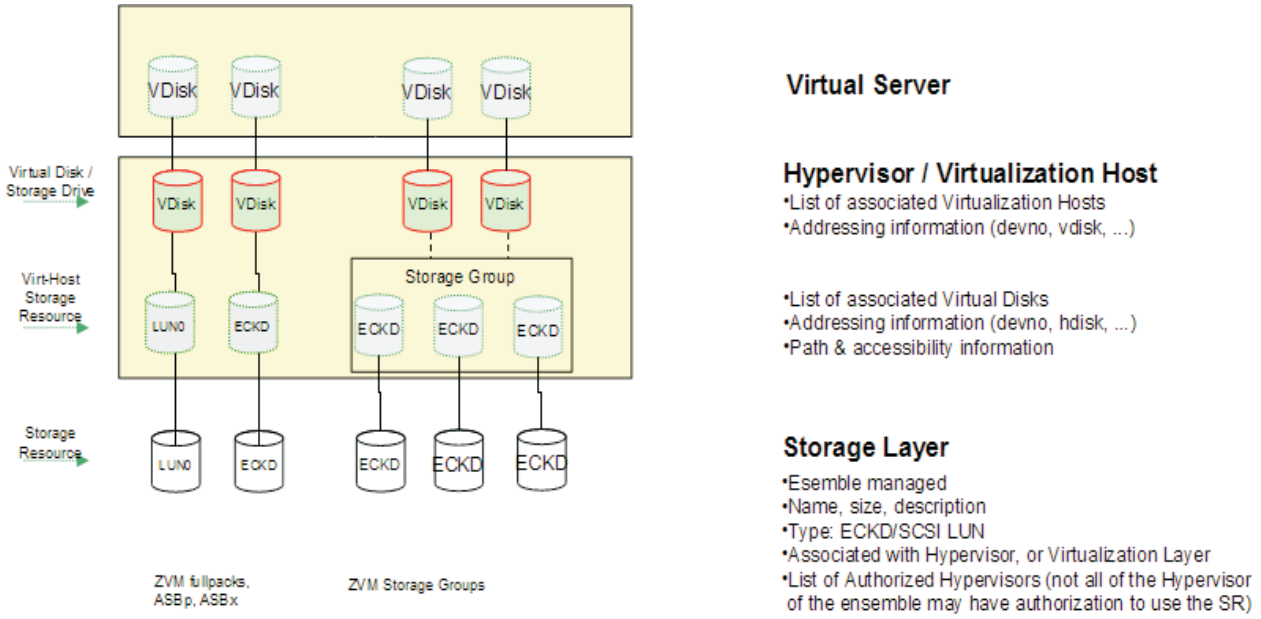

Figure 141. Object model

Storage management operations summary

The following tables provide an overview of the operations provided. The tables are organized according to the scope of the operations listed.

Table 55. Storage management: ensemble-level storage operations

Operation name	HTTP method and URI path
"List Storage Resources" on page 299	`GET /api/ensembles/{ensemble-id}/storage-resources`
"Get Storage Resource Properties" on page 302	`GET /api/storage-resources/{storage-resource-id}`
"Create Storage Resource" on page 303	`POST /api/ensembles/{ensemble-id}/storage-resources`

Table 55. Storage management: ensemble-level storage operations (continued)

Operation name	HTTP method and URI path
"Update Storage Resource Properties" on page 305	`POST /api/storage-resources/{storage-resource-id}`
"Delete Storage Resource" on page 307	`DELETE /api/storage-resources/{storage-resource-id}`
"Export World Wide Port Names List" on page 309	`POST /api/ensembles/{ensemble-id}/operations/export-port-names`
"Import Storage Access List" on page 311	`POST /api/ensembles/{ensemble-id}/operations/import-storage-access-list`

Table 56. Storage management: virtualization host storage operations

Operation name	HTTP method and URI path
"List Virtualization Host HBA Ports" on page 316	`GET /api/virtualization-hosts/{virt-host-id}/hba-ports`
"List Virtualization Host Storage Resources" on page 318	`GET /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-resources`
"Get Virtualization Host Storage Resource Properties" on page 321	`GET /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-resources/{virt-host-storage-resource-id}`
"Create Virtualization Host Storage Resource" on page 325	`POST /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-resources`
"Delete Virtualization Host Storage Resource" on page 328	`DELETE /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-resources/{virt-host-storage-resource-id}`
"Add Virtualization Host Storage Resource Paths" on page 330	`POST /api/virtualization-hosts/{virt-host-id}/operations/add-paths`
"Remove Virtualization Host Storage Resource Paths" on page 333	`POST /api/virtualization-hosts/{virt-host-id}/operations/remove-paths`
"Discover Virtualization Host Storage Resources" on page 336	`POST /api/virtualization-hosts/{virt-host-id}/operations/discover-virtualization-host-storage-resources`

Table 57. Storage management: storage group operations

Operation name	HTTP method and URI path
"List Virtualization Host Storage Groups" on page 339	`GET /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-groups`
"Get Virtualization Host Storage Group Properties" on page 342	`GET /api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-groups/{virt-host-storage-group-id}`
"List Virtualization Host Storage Resources in a Virtualization Host Storage Group" on page 344	`POST /api/virtualization-hosts/{virt-host-id}/operations/list-virtualization-host-storage-resources-in-group`
"Add Virtualization Host Storage Resource to Virtualization Host Storage Group" on page 346	`POST /api/virtualization-hosts/{virt-host-id}/operations/add-virtualization-host-storage-resource-to-group`
"Remove Virtualization Host Storage Resource from Virtualization Host Storage Group" on page 348	`POST /api/virtualization-hosts/{virt-host-id}/operations/remove-virtualization-host-storage-resource-from-group`

Table 58. Storage management: URI variables

Variable	Description
{ensemble-id}	Object ID of an ensemble object
{virt-host-id}	Object ID of a virtualization host object
{virtual-server-id}	Object ID of a virtual server object
{storage-resource-id}	Object ID of a storage resource object
{virt-host-storage-resource-id}	Element ID of a virtualization host storage resource object
{virt-host-storage-group-id}	Element ID of a virtualization host storage group object

Note: Although virtual disk operations are also storage related, they have a closer affinity to virtualization management. Thus, they are included within the specification for the "Virtual Server Object" on page 206 in Chapter 10, "Virtualization management," on page 161.

Storage resource object

A storage resource object represents a single physical storage resource available to one or more zEnterprise Virtualization Hosts in an ensemble.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, but does not provide the operational-status-related properties defined in that schema because it does not maintain the concept of an operational status.

The following type-specific specializations apply to the other Base Managed Object properties:

Table 59. Storage resource object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path for a storage resource object is of the form `/api/storage-resources/{storage-resource-id}` where `{storage-resource-id}` is the value of the object-id property of the storage resource object.
parent	—	String/ URI	The parent object of a storage resource object is an ensemble object.
class	—	String	The class of a storage resource object is **"storage-resource"**.
description	(w)(pc)	String (0-256)	The optional user-supplied description for the storage resource. This is the description that will be displayed on the user interface. It must consist only of alphanumeric characters, spaces and the following special characters: "._-".
name	(w)(pc)	String (1-64)	The user-supplied name of the storage resource. This is the name that will be displayed on the user interface. It must consist only of alphanumeric characters, spaces and the following special characters: "._-", and it must begin with an alphabetic character. This name must be unique within the Ensemble.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional type-specific properties:

Table 60. Storage resource object: class specific properties

Name	Qualifier	Type	Description
type	—	String Enum	The type of the storage resource. Values: • **"eckd"** – An Extended Count Key Data storage resource • **"fcp"** – A Fibre-channel attached storage resource • **"zvm-fcp"** – A Fibre-channel attached storage resource for use by z/VM virtualization hosts
size	(w)(pc)	Long	The **size** of the storage resource. The units for this property are specified by the **allocation-units** property.
allocation-units	(w)	String Enum	The units for the size property. Values: • **"bytes"**– used only for storage resources whose **type** property is **"fcp"** or **"zvm-fcp"**. • **"cylinders"** – used only for storage resources whose **type** property is **"eckd"**. • **"unknown"**
allocation-status	(pc)	String Enum	The status of the storage resource in terms of its current allocation. A storage resource is considered to be allocated for use if there is a virtual disk backed by this storage resource or the storage resource backs a hypervisor storage resource that is a member of a virtualization host storage group. Values: • **"free"** – the storage resource is not currently allocated for use. • **"used"** – the storage resource is currently allocated for use.
unique-device-id	(pc)	String	The unique device identifier assigned to this storage resource. This is a worldwide unique identifier based on information about the device. zManager creates a unique device identifier for each storage resource whose **type** property is **"fcp"**. A unique device identifier is not created for storage resources with a **type** property of **"eckd"** or **"zvm-fcp"**; for such storage resources, the value of this property is always null. Note that even for storage resources with a **type** property of **"fcp"**, the value of the property may be null or an empty string. This is typically the case if zManager has not yet accessed the storage resource and thus the unique device identifier is not yet known.

Operations

If a storage resource operation accesses a z/VM Virtualization Host and encounters an error while communicating with the Virtualization Host via SMAPI, the response body is as described in "SMAPI Error Response Body" on page 203.

List Storage Resources

The **List Storage Resources** operation lists the storage resources in the ensemble.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/storage-resources

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`storage-resources`	Array of objects	Array of storage-resource-basic-info objects, described in the next table. If no storage resources are to be returned, an empty array is provided.

Each storage-resource-basic-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the storage resource object
`name`	String	The **name** property of the storage resource object
`type`	String Enum	The **type** property of the storage resource object

Description

The **List Storage Resources** operation lists the storage resources in the ensemble. The object URI and other basic properties are provided for each storage resource.

If the **name** query parameter is specified, the returned list is limited to those storage resources that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

A set of basic properties is returned for each storage resource. See the storage-resource-basic-info object definition.

A storage resource is included in the list only if the API user has object-access permission for that object. If the ensemble contains a storage resource to which the API user does not have permission, that object is omitted from the list, but no error status code results. Note that this could result in an empty list.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

If there are no storage resources in the ensemble, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object specified in the request URI
- Object-access permission to the storage resource objects passed in the response body
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 300.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{ensemble-id}* does not designate an existing ensemble object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026/storage-resources HTTP/1.1
x-api-session: 1rmnds0imna61i3ll0eu7drk7jsec93mvclfbuqdb7xspk2fm5
```

Figure 142. List Storage Resources: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 04 Aug 2011 13:30:11 GMT
content-type: application/json;charset=UTF-8
content-length: 524
{
   "storage-resources": [
      {
         "name": "erictest",
         "object-uri": "/api/storage-resources/a23a998c-9693-11e0-aace-00215e69e3f5",
         "type": "fcp"
      },
      {
         "name": "Test2_ECKD",
         "object-uri": "/api/storage-resources/8d7da556-6598-11e0-8946-00215e69e3f5",
         "type": "eckd"
      },
      {
         "name": "FCP5618",
         "object-uri": "/api/storage-resources/b0be5b6e-5ada-11e0-b462-00215e69e3f5",
         "type": "zvm-fcp"
      }
   ]
}
```

Figure 143. List Storage Resources: Response

Get Storage Resource Properties

The **Get Storage Resource Properties** operation retrieves the properties of a single storage resource object that is designated by its object ID.

HTTP method and URI

GET /api/storage-resources/{*storage-resource-id*}

In this request, the URI variable *{storage-resource-id}* is the object ID of the storage resource object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the storage resource object as defined in the "Data model" on page 298. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get Storage Resource Properties** operation returns the current properties for the storage resource object specified by *{storage-resource-id}*.

On successful execution, all of the current properties as defined in "Data model" on page 298 for the storage resource object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing storage resource object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the storage resource object specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{storage-resource-id}* does not designate an existing storage resource object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334 HTTP/1.1
x-api-session: 1tcd8u2o682d6diyfqt8q9aafhfxl25d8m87l50yl6osfcje7k
```

Figure 144. Get Storage Resource Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 06 Dec 2011 16:00:26 GMT
content-type: application/json;charset=UTF-8
content-length: 402
{
   "allocation-status": "free",
   "allocation-units": "bytes",
   "class": "storage-resource",
   "description": "SS Ensemble volume V001",
   "is-locked": false,
   "name": "SS-V0001",
   "object-id": "6967f806-2023-11e1-9c1e-0010184c8334",
   "object-uri": "/api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334",
   "parent": "/api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334",
   "size": 8589934592,
   "type": "fcp",
   "unique-device-id": null
}
```

Figure 145. Get Storage Resource Properties: Response

Create Storage Resource

The **Create Storage Resource** operation adds a storage resource to the specified ensemble.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/storage-resources

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble to which the new storage resource is to be added.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
name	String	Required	The **name** property of the new storage resource
description	String	Optional	The **description** property of the new storage resource
size	Long	Required	The **size** property of the storage resource
type	String Enum	Required	The **type** property of the storage resource

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the storage resource object, in the form /api/storage-resources/*{storage-resource-id}*

Description

The **Create Storage Resource** operation identifies a new storage resource to be added to the ensemble specified by the *{ensemble-id}* portion of the request URI. Once added to the ensemble, the storage resource can be managed using the various storage-related zManager functions.

On successful execution, the **object-uri** field of the response body and the **Location** response header identify the new storage resource.

If this operation changes the value of any property for which property-change notifications are due, those notifications are issued asynchronously to this operation. Upon success, an Inventory Change notification is issued.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action access permission to the **Add Storage Resource** task; otherwise, status code 403 (Forbidden) is returned.

If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object specified in the request URI
- Action/task permission to the **Add Storage Resource** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and both the response body and the **Location** response header contain the URI of the newly created object.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	8	The name specified for the new storage resource is not unique within the ensemble.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{ensemble-id}* does not designate an existing ensemble object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

The **Create Storage Resource** operation performs a portion of the function provided by the Add Storage Resource subtask of the **Manage Storage Resources** task. The Add Storage Resource subtask performs some or all of the functions provided by the following Web Services APIs:

- **Create Storage Resource**

  Creates a storage resource object in zManager, representing storage resources (FCP LUNs, or ECKD™ volumes). Each storage resource object has a unique name. zManager reports additional storage resource specific information, such as a unique-device-id, after paths have been added and zManager was able to access the storage resource.
- **Create Virtualization Host Storage Resource**

  Creates an object in zManager, representing the virtualization host's view on the storage resource.
- **Add Virtualization Host Storage Resource Path**

  Adds a path between a virtualization host and a storage resource.

  Note: a path to an FCP storage resource is identified by a host-port-wwpn, target-port-wwpn, and lun. See Table 62 on page 315. A path to an ECKD resource is identified by the device number of the ECKD volume. See Table 63 on page 316.

Example HTTP interaction

```
POST /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/storage-resources HTTP/1.1
x-api-session: 1tcd8u2o682d6diyfqt8q9aafhfxl25d8m87l50yl6osfcje7k
content-type: application/json
content-length: 109
{
   "description": "New Storage Resource",
   "name": "SS-New-Storage-Resource",
   "size": 8589934592,
   "type": "fcp"
}
```

Figure 146. Create Storage Resource: Request

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334
cache-control: no-cache
date: Tue, 06 Dec 2011 16:00:26 GMT
content-type: application/json;charset=UTF-8
content-length: 76
{
   "object-uri": "/api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334"
}
```

Figure 147. Create Storage Resource: Response

Update Storage Resource Properties

The **Update Storage Resource Properties** operation updates one or more of the writeable properties of a storage resource object.

HTTP method and URI

POST /api/storage-resources/{*storage-resource-id*}

In this request, the URI variable *{storage-resource-id}* is the object ID of the storage resource object for which properties are to be updated.

Request body contents

The request body is expected to contain a JSON object that provides the new values of any writeable property that is to be updated by this operation. Field names and data types in this JSON object are expected to match the corresponding property names and data types defined in the data model for this object type. The JSON object can and should omit fields for properties whose values are not to be changed by this operation.

Description

The **Update Storage Resource Properties** operation updates writeable properties of the storage resource object specified by *{storage-resource-id}*.

The request body contains an object with one or more fields with field names that correspond to the names of properties for this object. On successful execution, the value of each corresponding property of the object is updated with the value provided by the input field, and status code 204 (No Content) is returned without supplying any response body. The request body does not need to specify a value for all writeable properties, but rather can and should contain fields for the properties to be updated. Object properties for which no input value is provided remain unchanged by this operation.

If the update changes the value of any property for which property-change notifications are due, those notifications are issued asynchronously to this operation.

The URI path must designate an existing storage resource object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Storage Resources Details** task; otherwise, status code 403 (Forbidden) is returned.

The request body is validated against the data model for this object type to ensure that it contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the storage resource object specified in the request URI
- Action/task permission to the **Storage Resources Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	8	The new name specified for the new storage resource is not unique within the ensemble.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{storage-resource-id}* does not designate an existing storage resource object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334 HTTP/1.1
x-api-session: 1tcd8u2o682d6diyfqt8q9aafhfxl25d8m87l50yl6osfcje7k
content-type: application/json
content-length: 62
{
   "description": "SS Ensemble volume V001",
   "name": "SS-V0001"
}
```

Figure 148. Update Storage Resource Properties: Request

```
204 No Content
date: Tue, 06 Dec 2011 16:00:26 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 149. Update Storage Resource Properties: Response

Delete Storage Resource

The **Delete Storage Resource** operation deletes the specified storage resource from the ensemble.

HTTP method and URI

DELETE /api/storage-resources/{*storage-resource-id*}

In this request, the URI variable *{storage-resource-id}* is the object ID of the storage resource object to be deleted.

Description

The **Delete Storage Resource** operation removes a specified storage resource from the ensemble. The storage resource is identified by the *{storage-resource-id}* variable in the URI. There must be no virtualization host storage resources associated with the storage resource to be deleted.

Upon successfully removing the storage resource, HTTP status code 204 (No Content) is returned and no response body is provided. An inventory change event is issued asynchronously.

The URI path must designate an existing storage resource object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Remove Storage Resource** task; otherwise, status code 403 (Forbidden) is returned. If there are any virtualization host storage resources associated with the storage resource, status code 409 (Conflict) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the storage resource object specified in the request URI
- Action/task permission to the **Remove Storage Resource** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{storage-resource-id}* does not designate an existing storage resource object, or the API user does not have object access permission to it.
409 (Conflict)	143	The object cannot be deleted at this time. There is a virtualization host storage resource associated with the storage resource.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- Before a storage resource can be deleted from the ensemble, any virtualization host storage resources associated with it must first be deleted.

Example HTTP interaction

```
DELETE /api/storage-resources/6967f806-2023-11e1-9c1e-0010184c8334 HTTP/1.1
x-api-session: 1tcd8u2o682d6diyfqt8q9aafhfxl25d8m87l50yl6osfcje7k
```

Figure 150. Delete Storage Resource: Request

```
204 No Content
date: Tue, 06 Dec 2011 16:00:26 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 151. Delete Storage Resource: Response

Export World Wide Port Names List

The **Export World Wide Port Names List** operation exports the world wide port names (WWPNs) of the fibre channel host ports of the specified virtualization hosts.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/operations/export-port-names

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble that contains the virtualization hosts specified in the request body.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-hosts`	Array of String/URI	Required	Array of canonical URI paths, one for each virtualization host whose WWPN list is to be exported

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`wwpn-list`	String	The WWPN list in Comma-Separated Values (CSV) format

Description

The **Export World Wide Port Names List** operation returns the list of host port WWPNs of the virtualization hosts specified by the **virtualization-hosts** field of the request body. These virtualization hosts must be part of the ensemble specified by *{ensemble-id}*. The list is provided in a JSON object as a single string in Comma-Separated Values (CSV) format. It will be of the format described in the **Import Storage Access List** operation, with only the statement type, **Location**, and **HostWwpn** fields filled in. This result can be used as the basis for a Storage Access List to be supplied as input to the **Import Storage Access List** operation.

On successful execution, the WWPN list for the specified virtualization hosts is provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. If the array of virtualization host URIs is empty, status code 400 (Bad Request) is returned. The URIs in the request body must designate existing virtualization host objects and the API user must have object-access

permission to them; otherwise, status code 404 (Not Found) is returned. The virtualization hosts must be part of the specified ensemble; otherwise, status code 400 (Bad Request) is returned. In addition, the API user must have action access permission to the **Export WWPNs** task; otherwise, status code 403 (Forbidden) is returned.

The request body is validated against the schema described in "Request body contents" on page 309. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object specified in the request URI
- Object-access permission to the hosting object of the virtualization hosts specified in the request body
- Action/task permission to the **Export WWPNs** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 309.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	146	A virtualization host specified in the request body is not a member of the ensemble specified in the request URI.
	149	The array of virtualization host URIs in the request body is empty.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{ensemble-id}* does not designate an existing ensemble object, or the API user does not have object access permission to it.
	2	A URI specified in the request body does not identify a virtualization host.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- This operation creates the same file as the **Export Host Port WWPN** task. This file can be used as the basis for a Storage Access List. The Comma-Separated Value format has been chosen because it allows customers to use spreadsheet applications to display, sort, add or modify data.

- The **List Virtualization Host HBA Ports** operation provides similar information in JSON format.

Example HTTP interaction

```
POST /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/operations/export-port-names HTTP/1.1
x-api-session: 5mkvfjvxt6guptdr5omvc11et4tvazeb684stvvhq4c1aa7w4x
content-type: application/json
content-length: 158
{
   "virtualization-hosts": [
      "/api/virtualization-hosts/71822c16-0401-11e1-8eda-001f163805d8",
      "/api/virtualization-hosts/2f676d90-03f8-11e1-8eda-001f163805d8"
   ]
}
```

Figure 152. Export World Wide Port Names List: WWPN list: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 06:47:31 GMT
content-type: application/json;charset=UTF-8
content-length: 394
{
   "wwpn-list":
       "#Version: 1
       #FCP_DEF:,Name,Size,Description,Location,HostWwpn,TargetWwpn,Lun
       #ECKD_DEF:,Name,Size,Description,Location,Devno,Volser
       #ZVM_FCP_DEF:,Name,Size,Description,Location,Devno,Volser,HostWwpn,TargetWwpn,Lun
       FCP,,,,R32:B.2.12,21000024ff24df01
       FCP,,,,R32:B.2.12,21000024ff24df00
       FCP,,,,R32:B.2.02,21000024ff2b47cb
       FCP,,,,R32:B.2.02,21000024ff2b47ca
       "
}
```

Figure 153. Export World Wide Port Names List: WWPN list: Response

Import Storage Access List

The **Import Storage Access List** operation imports information about storage resources and the virtualization hosts that have access to them. The Storage Access List (SAL) contains information, such as host port WWPNs and properties, such as addressing and device type information for configured storage resources.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/operations/import-storage-access-list

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object on which the Storage Access List is to be imported.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
sal	String	Required	Storage Access List in Comma-Separated Values (CSV) format

The largest request body accepted by this operation is 1 MB. Requests with bodies that exceed this maximum are rejected with an HTTP status 413 (Request Entity Too Large) response.

Description

The **Import Storage Access List** operation imports the provided Storage Access List into the ensemble specified by *{ensemble-id}*. The Storage Access List specifies paths between host ports and configured storage resources. It is a convenient way to specify which virtualization hosts have access to each storage resource.

The Storage Access List logically consists of lines of text, each one being a statement that identifies a storage resource, a virtualization host that has access to that storage resource and a path for the virtualization host to use when accessing the storage resource. For the full definition of the Storage Access List, see the storage access list worksheet described in the appendix in the *zEnterprise System Ensemble Planning and Configuring Guide*.

On successful execution, status code 204 (No Content) is returned without supplying a response body.

The URI path must designate an existing ensemble object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Import SAL** task; otherwise, status code 403 (Forbidden) is returned. All virtualization hosts designated in the Storage Access List are marked busy for the duration of this request. If any of those virtualization hosts is already marked busy due to some other operation, then status code 409 (Conflict) is returned.

The request body is validated against the schema described in "Request body contents" on page 311. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered. In addition, the CSV-formatted SAL designated by the **sal** field must be syntactically and semantically correct; otherwise, status code 400 (Bad Request) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object specified in the request URI
- Action/task permission to the **Import SAL** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	140	The CSV-formatted Storage Access List designated by the **sal** field is not syntactically and semantically correct, or an error was encountered while processing a Storage Access List entry. The response body contains a message with more details.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{ensemble-id}* does not designate an existing ensemble object, or the API user does not have object access permission to it.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because a virtualization host designated by the Storage Access List is currently busy performing some other operation.
	150	The operation cannot be performed because a virtualization host designated by the Storage Access List is currently busy due to a zBX Move operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Usage notes

- This operation accepts the same data / file as the **Import Storage Access List** task. This file contains information about storage resources, virtualization host storage resources, and path information that is to be added to zManager.

  The Comma-Separated Value format has been chosen because it allows customers to use spreadsheet applications in order to display, sort, add or modify data.
- The **Create Storage Resource**, **Create Virtualization Host Storage Resource**, and **Add Virtualization Host Storage Resource Paths** operations can be used to provide the same information to zManager in JSON format.

Inventory service data

Information about the storage resources managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for Storage Resource objects are included in the response to the Inventory Service's Get Inventory operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"storage-resource"** are to be included. An entry for a particular storage resource is included only if the API user has object-access permission to that object.

For each Storage Resource object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for the Get Storage Resource Properties operation. That is, the data provided is the same as would be provided if a Get Storage Resource Properties operation were requested targeting this object.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the Get Inventory response to describe a single storage resource. This object would appear as one array entry in the response array:

```
{
   "allocation-status": "used",
   "allocation-units": "bytes",
   "class": "storage-resource",
   "description": "2024-0080-e518-4ac0\r\n0000-0000-0000-0000",
   "name": "B1010000A",
   "object-id": "2d588ee2-25a2-11e0-94a7-0010184c8334",
   "object-uri": "/api/storage-resources/2d588ee2-25a2-11e0-94a7-0010184c8334",
   "parent": "/api/ensembles/87d73ffc-75b2-11e0-9ba3-0010184c8026",
   "size": 10737418240,
   "type": "fcp",
   "unique-device-id": "3E21360080E5000184AC00000441B4D07449B0F1814 FAStT03IBMfcp"
}
```

Figure 154. Storage resource object: Sample inventory data

Virtualization host storage resource object

A virtualization host storage resource represents a storage resource to which the virtualization host has been granted access. It is the representation of a storage resource from the perspective of a virtualization host.

Data model

This object includes the following properties:

Table 61. Virtualization host storage resource object properties

Name	Qualifier	Type	Description
element-uri	—	String/ URI	Canonical URI path of the virtualization host storage resource object, in the form `/api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-resources/{virt-host-storage-resource-id}` where *{virt-host-storage-resource-id}* is the value of the **element-id** property of this object.
element-id	—	String (36)	The unique identifier for the virtualization host storage resource instance. This identifier is in the form of a UUID.
parent	—	String/URI	The parent object of a virtualization host storage resource object is a virtualization host object.
class	—	String	The class of a virtualization host storage resource object is **"virtualization-host-storage-resource"**.
name	(pc)	String	The name of the storage resource, as defined in the Storage Resource object's Data Model section.
description	(pc)	String	The description for the storage resource, as defined in the Storage Resource object's Data Model section.
size	(pc)	Long	The size of the storage resource, as defined in the Storage Resource object's Data Model section.
allocation-units	—	String Enum	The units for the **size** property, as defined in the Storage Resource object's "Data model" on page 298.
type	—	String Enum	The type of the storage resource, as defined in the Storage Resource object's Data Model section.
storage-resource	—	String/ URI	Canonical URI path of the storage resource object associated with this virtualization host storage resource.

Table 61. Virtualization host storage resource object properties (continued)

Name	Qualifier	Type	Description
paths	(w)	Array of objects	Information about the paths by which the storage resource is accessible to this virtualization host. It is an array of path-information-fcp or path-information-eckd objects. If the virtualization host has no paths to the storage resource, an empty array is provided. Note that there can be at most one path for a virtualization host storage resource whose **type** property is **"eckd"**.
unique-device-id	(pc)	String	The unique device identifier of the storage resource, as defined in the Storage Resource object data model. See "Class specific additional properties" on page 298.
volume-serial-number	(pc)	String (1-6)	The volume serial for this virtualization host storage resource. Only present if the **type** property is **"eckd"** or **"zvm-fcp"**.
device-number	—	String (1-4)	The device number that the virtualization host uses to access an ECKD storage resource, or the EDEV (emulated volume) that is created for FCP storage resources when they get allocated to a Virtual Server, or added to a Virtualization Host Storage Group. The string form of a 1-4 digit hexadecimal number. Only present if the **type** property is **"eckd"** or **"zvm-fcp"**.
virtualization-host-storage-group	(pc)	String/URI	Canonical URI path of the virtualization host storage group of which this virtualization host storage resource is a member or null if this virtualization host storage resource is not in a virtualization host storage group. Only present if the **type** property is **"eckd"** or **"zvm-fcp"**.

A path-information-fcp object contains information about a single path by which an FCP storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is **"fcp"** or **"zvm-fcp"**.

Table 62. Virtualization host storage resource object: path-information-fcp object properties

Name	Type	Description
host-port-wwpn	String (16)	The WWPN of the host port. The string form of a 16-digit hexadecimal number.
controller-port-wwpn	String (16)	The WWPN of the storage controller port. The string form of a 16-digit hexadecimal number.
lun	String (16)	The Logical Unit Number (LUN) of the storage resource. The string form of a 16-digit hexadecimal number.
accessible	Boolean	Indicates whether the storage resource is currently accessible to the virtualization host via this path. Because path accessibility status can be time consuming to determine, by default such status is omitted when Virtualization Host Storage Resource properties are returned and instead the value provided for this property is **null**. Operations that provide path accessibility status data will specifically indicate the conditions under which they do so.

A path-information-eckd object contains information about a single path by which an ECKD storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is **"eckd"**.

Note: the path specified through URM is the device number specified in the system I/O Configuration for the ECKD volume. It is not to be confused with the path (CHPID) between the system and the Control Unit.

Table 63. Virtualization host storage resource object: path-information-eckd object properties

Name	Type	Description
device-number	String (1-4)	The device number of the storage resource. The string form of a 1-4 digit hexadecimal number.
accessible	Boolean	Indicates whether the storage resource is currently accessible to the virtualization host via this path. Because path accessibility status can be time consuming to determine, by default such status is omitted when Virtualization Host Storage Resource properties are returned and instead the value provided for this property is **null**. Operations that provide path accessibility status data will specifically indicate the conditions under which they do so.

Operations

If a virtualization host storage resource operation accesses a z/VM virtualization host and encounters an error while communicating with the virtualization host via SMAPI, the response body is as described in "SMAPI Error Response Body" on page 203.

List Virtualization Host HBA Ports

The **List Virtualization Host HBA Ports** operation lists information about Fibre-Channel HBA (Host Bus Adapter) ports for a virtualization host.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/hba-ports

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`hba-ports`	Array of objects	Information about the virtualization host's HBA ports. It is an array of hba-port-information objects, described in the next table. If no virtualization host HBA ports are to be returned, an empty array is provided.

Each hba-port-information object contains the following fields:

Field name	Type	Description
`wwpn`	String	World Wide Port Name (WWPN) of the port.
`identifier`	String	Identifier for the port. It contains the device number of the subchannel for a virtualization host whose **type** property is **"zvm"**. For Power ASB, an example is fscsi2. For Intel based ASBs, the identifier starts with "/dev/...".

Description

The **List Virtualization Host HBA Ports** operation lists a virtualization host's HBA ports. All properties are provided for each port.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

If there are no HBA ports for the virtualization host, an empty list is provided and the operation completes successfully with HTTP status code 200 (OK).

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Export WWPNs** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 316.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de/hba-ports HTTP/1.1
x-api-session: 1rmnds0imna61i3ll0eu7drk7jsec93mvclfbuqdb7xspk2fm5
```

Figure 155. List Virtualization Host HBA Ports: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 04 Aug 2011 13:30:12 GMT
content-type: application/json;charset=UTF-8
content-length: 210
{
   "hba-ports": [
      {
         "identifier": "fscsi3",
         "wwpn": "21000024ff2b47cb"
      },
      {
         "identifier": "fscsi2",
         "wwpn": "21000024ff2b47ca"
      }
   ]
}
```

Figure 156. List Virtualization Host HBA Ports: Response

List Virtualization Host Storage Resources

The **List Virtualization Host Storage Resources** operation lists the virtualization host storage resources for a virtualization host.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-resources

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host.

Query parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property
properties	String	Optional	Identifies the properties of each virtualization host storage resource to be returned. The only supported value is **"all"**, which results in all properties being returned, including path accessibility status properties. If this query parameter is omitted, a set of basic properties is returned.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
virtualization-host-storage-resources	Array of objects	If the **properties=all** query parameter is specified, an array is provided whose elements are the set of virtualization host storage resource properties that would be returned on a **Get Virtualization Host Storage Resource Properties** request with the **include-path-accessibility** query parameter specified as true. If the **properties** query parameter is omitted, an array of virtualization-host-storage-resource-basic-info objects is returned, described in the next table. If no virtualization host storage resources are to be returned, an empty array is provided.

Each virtualization-host-storage-resource-basic-info object contains the following fields:

Field name	Type	Description
element-URI	String/ URI	Canonical URI path of the virtualization host storage resource object
name	String	The **name** property of the associated storage resource object
type	String Enum	The **type** property of the associated storage resource object

Description

The **List Virtualization Host Storage Resources** operation lists a virtualization host's virtualization host storage resources. The object URI and other basic properties are provided for each virtualization host storage resource.

If the **name** query parameter is specified, the returned list is limited to those virtualization host storage resources that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

If the **properties** query parameter is specified, it controls the set of properties returned. A value of **"all"** results in all properties being returned, in exactly the same format as would be provided on a **Get Virtualization Host Storage Resource Properties** request with the **include-path-accessibility** query parameter specified as true. If the **properties** query parameter is omitted, a set of basic properties is returned for each virtualization host storage resource. See the virtualization-host-storage-resource-basic-info object definition. Any value other than **all** is not valid and results in an HTTP status code 400 (Bad Request).

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

The virtualization host designated by the URI path must have a **status** of **"operating"**, otherwise status code 409 (Conflict) is returned.

If there are no virtualization host storage resources for the virtualization host, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 318.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	145	A value other than **"all"** was specified for the **properties** query parameter, or this query parameter was specified more than once.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.
409 (Conflict)	1	Virtualization host has a **status** that is not valid for this operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- The information provided about a virtualization host storage resource can optionally include path accessibility information, i.e. information on whether a path to the storage resources is currently operational or not. However, determining path accessibility status can be an expensive action, especially when a virtualization host has one or more non-functional paths as SAN rediscovery is implicitly triggered in this case. For this reason, an API application should obtain path accessibility information only if it is required to satisfy the function of the application. If not required, the application should obtain virtualization host storage resource information using techniques that omit path accessibility information in order to avoid unnecessary delays.

  If path-accessibility information is required for all or many virtualization host storage resources, it may be significantly faster to obtain that information by making a single request to this operation with the **properties=all** parameter specified rather than by making a series of requests to the **Get Virtualization Host Storage Resource Properties** operation (with the **include-path-accessibility=true** parameter specified) to obtain hat information one resource at a time. This is because overhead due to SAN rediscovery would be incurred at most one time by using this operation, but might be incurred on each and ever iterated **Get Virtualization Host Storage Resource Properties** request.

  On the other hand, if path-accessibility information is not required, using this operation with the **properties=all** parameter specified may incur unnecessary application delays. Instead, the application can bypass the determination of path-accessibility status by using this operation with **properties=all** omitted to obtain the URIs of the virtualization host's storage resources, and then iterating over those URIs and making a requests to the **Get Virtualization Host Storage Resource Properties** operation with **include-path-accessibility=false** specified (or defaulted) for each. If the application requires virtualization host storage resource information for all or many virtualization hosts in the ensemble, obtaining this information via the **Get Inventory** operation of the Inventory Service (for the virtualization host inventory categories) may provide the best performance. This service cab provide data across all virtualization hosts in the ensemble in a single request, and bypasses the potentially costly determination of path-accessibility status when obtaining storage resource information.

Example HTTP interaction

```
GET /api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de/virtualization-host-
  storage-resources HTTP/1.1
x-api-session: 1rmnds0imna61i3ll0eu7drk7jsec93mvclfbuqdb7xspk2fm5
```

Figure 157. List Virtualization Host Storage Resources: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 04 Aug 2011 13:30:12 GMT
content-type: application/json;charset=UTF-8
content-length: 729
{
"virtualization-host-storage-resources": [
{
"element-uri": "/api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de/virtualization-
  host-storage-resources/dfbf23f0-b7b7-11e0-a46d-f0def152d359",
"name": "B2L0012",
"type": "fcp"
},
{
"element-uri": "/api/virtualization-hosts/ba97ff30-2990-11e0-8d5b-001f163803de/virtualization-
  host-storage-resources/ff83dea8-64d1-11e0-b579-f0def10c03f4",
"name": "test1234",
"type": "fcp"
}
]
}
```

Figure 158. List Virtualization Host Storage Resources: Response

Get Virtualization Host Storage Resource Properties

The **Get Virtualization Host Storage Resource Properties** operation retrieves the properties of a single virtualization host storage resource object.

HTTP method and URI

**GET /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-resources/
  {*virt-host-storage-resource-id*}**

URI variables

Variable	Description
{virt-host-id}	Object ID of the virtualization host
{virt-host-storage-resource-id}	Element ID of the virtualization host storage resource object for which properties are to be obtained

Query parameters

Name	Type	Rqd/Opt	Description
`include-path-accessibility`	Boolean	Optional	If specified as true, the accessibility status of the paths for this resource is determined and reported as values of the **accessible** property. If specified as false or omitted, this status is not determined and instead the value of the **accessible** property is always **null**.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the virtualization host storage resource object as defined in "Data model" on page 314. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get Virtualization Host Storage Resource Properties** operation returns the current properties for the virtualization host storage resource object that is specified by its object ID *{virt-host-id}* and the object ID of the owning virtualization host *{virt-host-storage-resource-id}*.

On successful execution, all of the current properties as defined in "Data model" on page 314 for the virtualization host storage resource object are provided in the response body and HTTP status code 200 (OK) is returned. If the **include-path-accessibility** query parameter is specified as true, these properties include the current path accessibility status of each of the paths for the resource (as the **accessible** property). If **include-path-accessibility** is false (which is the default), this path accessibility status is not provided and instead the **accessible** property is always null.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtualization host storage resource object. If any of these conditions are not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

The virtualization host designated by the URI path must have a **status** of **"operating"**, otherwise status code 409 (Conflict) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.
	147	The object ID *{virt-host-storage-resource-id}* does not designate an existing virtualization host storage resource object for the specified virtualization host.
	148	There is no storage resource object associated with the virtualization host storage resource object identified by the object ID *{virt-host-storage-resource-id}*. This is most likely a temporary condition due to a delete operation in progress on the storage resource object.
409 (Conflict)	1	Virtualization host has a **status** that is not valid for this operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- Determining path accessibility status can be an expensive action, especially when a virtualization host has one or more non-functional paths as SAN rediscovery is implicitly triggered in this case. For this reason, this operation omits path accessibility status information (the **accessible** property) by default. Use the **include-path-accessibility** query parameter to request that this status be determined and reported when specifically needed by the application.
- If an application requires path accessibility status information for all or many of the storage resources of a virtualization host, IBM recommends using the **List Virtualization Host Storage Resources** operation with the **properties**=**all** query parameter to obtain information for all resources in a single request as a better performing approach than iteratively using **Get Virtualization Host Storage Resources** (with **include-path-accessibility**=**true**) one resource at a time. Repeated use of Get Virtualization Host Storage Resource may incur SAN rediscovery overhead once per request, but such overhead would be incurred at most once in the single List Virtualization Host of Storage Resources request.

Example HTTP interaction

```
GET /api/virtualization-hosts/75ca2d2e-e854-11df-811c-00262df32766/
  virtualization-host-storage-resources/c0261be8-ec51-11df-85fe-00262df32766 HTTP/1.1
x-api-session: 3gcd77glemvwq81dlmwxc8i4fwm4udxlby6i2auls4r6g529p1
```

Figure 159. Get Virtualization Host Storage Resource Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 06 Dec 2011 14:34:00 GMT
content-type: application/json;charset=UTF-8
content-length: 1043
{
   "allocation-units": "bytes",
   "class": "virtualization-host-storage-resource",
   "description": "xiv-41cb",
   "element-id": "c0261be8-ec51-11df-85fe-00262df32766",
   "element-uri": "/api/virtualization-hosts/75ca2d2e-e854-11df-811c-00262df32766/
    virtualization-host-storage-resources/c0261be8-ec51-11df-85fe-00262df32766",
   "name": "r93c1_12_hdisk4",
   "parent": "/api/virtualization-hosts/75ca2d2e-e854-11df-811c-00262df32766",
   "paths": [
      {
         "accessible": null,
         "controller-port-wwpn": "500173800aa50180",
         "host-port-wwpn": "2101001b32bf37e3",
         "lun": "41cb000000000000"
      },
      {
         "accessible": null,
         "controller-port-wwpn": "500173800aa50142",
         "host-port-wwpn": "2100001b329f37e3",
         "lun": "41cb000000000000"
      }
   ],
   "size": 34359738368,
   "storage-resource": "/api/storage-resources/c04f6f70-ec51-11df-a5bc-00215e6a0c27",
   "type": "fcp",
   "unique-device-id": "26112001738000AA5010B072810XIV03IBMfcp"
}
```

Figure 160. Get Virtualization Host Storage Resource Properties: Response for Virtualization Host of type "power-vm" or "x-hyp"

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 06 Dec 2011 14:34:01 GMT
content-type: application/json;charset=UTF-8
content-length: 651
{
   "allocation-units": "cylinders",
   "class": "virtualization-host-storage-resource",
   "description": "B71C",
   "device-number": "B71C",
   "element-id": "e85e91c2-ee02-11e0-a0eb-00262df332b3",
   "element-uri": "/api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3/
    virtualization-host-storage-resources/e85e91c2-ee02-11e0-a0eb-00262df332b3",
   "name": "B71C",
   "parent": "/api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3",
   "paths": [
      {
         "accessible": null,
         "device-number": "B71C"
      }
   ],
   "size": 60000,
   "storage-resource": "/api/storage-resources/ec9c3852-ee02-11e0-bc09-00215e6a0c27",
   "type": "eckd",
   "virtualization-host-storage-group": null,
   "volume-serial-number": "NNB7BC"
}
```

Figure 161. Get Virtualization Host Storage Resource Properties: Response for Virtualization Host of type "zvm"

Create Virtualization Host Storage Resource

The **Create Virtualization Host Storage Resource** operation creates a new virtualization host storage resource for the virtualization host.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-resources

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host that owns the new/modified virtualization host storage resource.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`storage-resource`	String/URI	Required	Canonical URI path of the storage resource object to be associated with this virtualization host storage resource.
`paths`	Array of objects	Optional	The path information for the new virtualization host storage resource. It is either an array of new-path-fcp or new-path-eckd objects, as described in the **Add Virtualization Host Storage Resource Path** operation. Note that there can be at most one path for a virtualization host storage resource whose **type** property is **"eckd"**.

Field name	Type	Rqd/Opt	Description
`volume-serial-number`	String (1-6)	Required when creating a VHSR on z/VM	The **volume-serial-number** property of the virtualization host storage resource object. This field is required when this operation is to create a new virtualization host storage resource on z/VM; otherwise, it is optional. This field only applies to storage resources whose **type** property is **"eckd"**or **"zvm-fcp"**. The volume serial number is written to the ECKD storage resource or FCP storage resources when they get added to a virtualization host storage group. The same volume serial number is to be used when creating multiple virtualization host storage resources for a storage resource.
`device-number`	String (1-4)	Required when creating an FCP VHSR on z/VM	The **device-number** property of the virtualization host storage resource object. This field is required when this operation is to create a new virtualization host storage resource; otherwise, it is optional. This field only applies to storage resources whose **type** property is **"eckd"**or **"zvm-fcp"**. The device number is used to access an ECKD storage resource, or assigned to the EDEV (emulated volume) that is created for FCP storage resources when they get allocated to a virtual server, or added to a virtualization host storage group.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the virtualization host storage resource object, in the form /api/virtualization-hosts/*{virt-host-id}*/virtualization-host-storage-resources/*{virt-host-storage-resource-id}*

Description

The **Create Virtualization Host Storage Resource** operation creates a new virtualization host storage resource for the virtualization host specified by the *{virt-host-id}* portion of the request URI.

Upon successful completion, the **element-uri** field of the response body and the **Location** response header identify the new virtualization host storage resource. An inventory change event is emitted asynchronously. See “Notifications” on page 338 for more information.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Add Storage Resource** task; otherwise, status code 403 (Forbidden) is returned. The virtualization host is marked busy for the duration of this request. If it is already marked busy due to some other operation, then status code 409 (Conflict) is returned.

If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Add Storage Resource** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and both the response body and the **Location** response header contain the URI of the newly created object.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	7	The storage resource object URI (**storage-resource**) in the request body designates a storage resource that is not compatible with the virtualization host designated in the request URI (*{virt-host-id}*).
	141	The virtualization host storage resource already has the maximum allowed number of paths.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.
	2	The storage resource URI specified in the request body does not identify a storage resource.
409 (Conflict)	2	The operation cannot be performed because a virtualization host designated by the request URI is currently busy performing some other operation.
	150	The operation cannot be performed because a virtualization host designated by the request URI is currently busy due to a zBX Move operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

The **Create Virtualization Host Storage Resource** operation performs a portion of the function provided by the Add Storage Resource subtask of the **Manage Storage Resources** task. The Add Storage Resource subtask performs some or all of the functions provided by the following Web Services APIs:

- **Create Storage Resource**

  Creates a storage resource object in zManager, representing storage resources (FCP LUNs, or ECKD volumes). Each storage resource object has a unique name. zManager reports additional storage resource specific information, such as a unique-device-id, after paths have been added and zManager was able to access the storage resource.
- **Create Virtualization Host Storage Resource**

  Creates an object in zManager, representing the hypervisor's view on the storage resource.
- **Add Virtualization Host Storage Resource Path**

  Adds a path between a virtualization host and a storage resource.

Example HTTP interaction

```
POST /api/virtualization-hosts/2f029af0-03f8-11e1-8eda-001f163805d8/
   virtualization-host-storage-resources HTTP/1.1
x-api-session: 68ps5rqvq1177xtcqz9rnrbmls29mglluq7ni0qlgnwjhup01c
content-type: application/json
content-length: 205
{
   "paths": [
      {
         "controller-port-wwpn": "20240080e5184ac0",
         "host-port-wwpn": "21000024ff2b4602",
         "lun": "1234001000000000"
      }
   ],
   "storage-resource": "/api/storage-resources/17556bdc-2034-11e1-83b8-0010184c8334"
}
```

Figure 162. Create Virtualization Host Storage Resource: Request

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/virtualization-hosts/2f029af0-03f8-11e1-8eda-001f163805d8/
   virtualization-host-storage-resources/19625098-2034-11e1-b4a5-001f163805d8
cache-control: no-cache
date: Tue, 06 Dec 2011 18:00:24 GMT
content-type: application/json;charset=UTF-8
content-length: 155
{
   "element-uri": "/api/virtualization-hosts/2f029af0-03f8-11e1-8eda-001f163805d8/
   virtualization-host-storage-resources/19625098-2034-11e1-b4a5-001f163805d8"
}
```

Figure 163. Create Virtualization Host Storage Resource: Response

Delete Virtualization Host Storage Resource

The **Delete Virtualization Host Storage Resource** operation removes a specified virtualization host storage resource from the specified virtualization host.

HTTP method and URI

DELETE /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-resources/
{*virt-host-storage-resource-id*}

URI variables

Variable	Description
{virt-host-id}	Object ID of the virtualization host
{virt-host-storage-resource-id}	Element ID of the virtualization host storage resource object to be deleted

Description

The **Delete Virtualization Host Storage Resource** operation removes a specified virtualization host storage resource from the specified virtualization host, removing all related path information at the same time. The virtualization host storage resource is identified by *{virt-host-storage-resource-id}* in the URI, and

the virtualization host is identified by *{virt-host-id}* in the URI. The virtualization host storage resource must not be part of a virtualization host storage group, and there must be no virtual disks backed by the virtualization host storage resource.

Upon successfully removing the virtualization host storage resource, HTTP status code 204 (No Content) is returned and no response body is provided. An inventory change event is issued asynchronously. See "Notifications" on page 338 for more information.

The URI path must designate an existing virtualization host and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtualization host storage resource object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must also have action access permission to the **Remove Storage Resource** task as well; otherwise, status code 403 (Forbidden) is returned. If the virtualization host storage resource is part of a virtualization host storage group or there is a virtual disk backed by the virtualization host storage resource, then status code 409 (Conflict) is returned. The virtualization host is marked busy for the duration of the request. If it is already marked busy due to some other operation, then status code 409 (Conflict) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Remove Storage Resource** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.
	147	The object ID in the URI *{virt-host-storage-resource-id}* does not designate an existing virtualization host storage resource object for the specified virtualization host.
409 (Conflict)	2	The operation cannot be performed because the virtualization host designated by the request URI is currently busy performing some other operation.
	144	The object cannot be deleted at this time. Either the virtualization host storage resource is part of a virtualization host storage group or a virtual disk is backed by the virtualization host storage resource.
	150	The operation cannot be performed because the virtualization host designated by the request URI is currently busy due to a zBX Move operation.

HTTP error status code	Reason code	Description
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage note

- While this operation does delete all path information associated with this virtualization host storage resource, it does not delete the associated storage resource, even if there is no longer a virtualization host storage resource associated with the storage resource. If the storage resource is no longer needed, it can be deleted using the **Delete Storage Resource** operation.

Example HTTP interaction

```
DELETE /api/virtualization-hosts/2f029af0-03f8-11e1-8eda-001f163805d8/
  virtualization-host-storage-resources/19625098-2034-11e1-b4a5-001f163805d8 HTTP/1.1
x-api-session: 68ps5rqvq1177xtcqz9rnrbmls29mglluq7ni0qlgnwjhup01c
```

Figure 164. Delete Virtualization Host Storage Resource: Request

```
204 No Content
date: Tue, 06 Dec 2011 18:00:53 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 165. Delete Virtualization Host Storage Resource: Response

Add Virtualization Host Storage Resource Paths

The **Add Virtualization Host Storage Resource Paths** operation adds a path definition to the virtualization host storage resource. The path is defined by its two endpoints – the virtualization host is at one end, and the associated storage resource is at the other end.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/operations/add-paths

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host object.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-host-storage-resource-element-uri`	String/URI	Required	Canonical URI path of the virtualization host storage resource object.
`paths`	Array of objects	Required	Either an array of new-path-fcp or new-path-eckd objects, described in the next table. Note that there can be at most one path for a virtualization host storage resource whose **type** property is **"eckd"**.

A new-path-fcp object contains information about a single path by which an FCP storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is either **"fcp"**or **"zvm-fcp"**.

Name	Type	Rqd/Opt	Description
`host-port-wwpn`	String (16)	Required	The WWPN of the host port. The string form of a 16-digit hexadecimal number.
`controller-port-wwpn`	String (16)	Required	The WWPN of the storage controller port. The string form of a 16-digit hexadecimal number.
`lun`	String (16)	Required	The Logical Unit Number (LUN) of the storage resource. The string form of a 16-digit hexadecimal number.

A new-path-eckd object contains information about a single path by which an ECKD storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is **"eckd"**.

Name	Type	Rqd/Opt	Description
`device-number`	String (1-4)	Required	The device number of the storage resource. The string form of a 1-4 digit hexadecimal number.

Description

The **Add Virtualization Host Storage Resource Paths** operation adds path definitions to the specified virtualization host storage resource. The **controller-port-wwpn**, **lun**, or **device-number**, as appropriate, along with the **host-port-wwpn**, must identify a configured path of the storage resource associated with this virtualization host storage resource. If that condition is not met, HTTP status code 400 (Bad Request) is returned.

Only a single path may be configured for an ECKD virtualization host storage resource. An attempt to define more than one path for such a resource will result in HTTP status code 400 (Bad Request) being returned.

The URI path must designate an existing virtualization host object, and the API user must have object-access permission to it. Furthermore, the request body must designate an existing virtualization host storage resource object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Add Storage Resource** task; otherwise, status code 403 (Forbidden) is returned. The virtualization host is marked busy for the duration of this request. If it is already marked busy due to some other operation, then status code 409 (Conflict) is returned.

Upon success, HTTP status code 204 (No Content) is returned and no response body is provided. If this operation changes the value of the **path** property, a property-change notification is issued asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Add Storage Resource** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	141	The virtualization host storage resource already has the maximum allowed number of paths.
	142	The specified path information does not identify a configured path available for the virtualization host to use to access the specified storage resource.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID in the URI (*{virt-host-id}*) does not designate an existing virtualization host object, or the API user does not have object access permission to it.
	147	The URI in the request body (*{virtualization-host-storage-resource}*) does not designate an existing virtualization host storage resource object for the specified virtualization host.
409 (Conflict)	2	The operation cannot be performed because the virtualization host designated by the request URI is currently busy performing some other operation.
	150	The operation cannot be performed because the virtualization host designated by the request URI is currently busy due to a zBX Move operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- The **Add Virtualization Host Storage Resource Paths** operation performs a portion of the function provided by the Add Storage Resource subtask of the **Manage Storage Resources** task. The Add Storage Resource subtask performs some or all of the functions provided by the following Web Services APIs:
  - **Create Storage Resource**

    Creates a storage resource object in zManager, representing storage resources (FCP LUNs, or ECKD volumes). Each storage resource object has a unique name. zManager reports additional storage resource specific information, such as a unique-device-id, after paths have been added and zManager was able to access the storage resource.
  - **Create Virtualization Host Storage Resource**

    Creates an object in zManager, representing the hypervisor's view on the storage resource.
  - **Add Virtualization Host Storage Resource Path**

    Adds a path between a virtualization host and a storage resource.
- There is no request to remove a virtualization host storage resource path. In order to remove a path, the virtualization host storage resource must be deleted and recreated with only the desired path(s).

Example HTTP interaction

```
POST /api/virtualization-hosts/2f029af0-03f8-11e1-8eda-001f163805d8/operations/add-paths HTTP/1.1
x-api-session: 68ps5rqvq1177xtcqz9rnrbmls29mglluq7ni0qlgnwjhup01c
content-type: application/json
content-length: 315
{
   "paths": [
      {
         "controller-port-wwpn": "20240080e5184ac0",
         "host-port-wwpn": "21000024ff2b4603",
         "lun": "1234001000000000"
      }
   ],
   "virtualization-host-storage-resource-element-uri": "/api/virtualization-hosts/
     2f029af0-03f8-11e1-8eda-001f163805d8/virtualization-host-storage-resources/
       19625098-2034-11e1-b4a5-001f163805d8"
}
```

Figure 166. Add Virtualization Host Storage Resource Paths: Request

```
204 No Content
date: Tue, 06 Dec 2011 18:00:53 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 167. Add Virtualization Host Storage Resource Paths: Response

Remove Virtualization Host Storage Resource Paths

The **Remove Virtualization Host Storage Resource Paths** operation removes a path definition from the virtualization host storage resource. The path is defined by its two endpoints – the virtualization host is at one end, and the associated storage resource is at the other end.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/operations/remove-paths

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host object.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-host-storage-resource-element-uri`	String/URI	Required	Canonical URI path of the virtualization host storage resource object.
`paths`	Array of objects	Required	Information about the paths to be removed. It is either an array of path-fcp or path-eckd objects, as described in the next tables. Note that there can be at most one path for a virtualization host storage resource whose **type** property is **"eckd"**.

A path-fcp object contains information about a single path by which an FCP storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is either **"fcp"**or **"zvm-fcp"**.

Name	Type	Rqd/Opt	Description
`host-port-wwpn`	String (16)	Required	The WWPN of the host port. The string form of a 16-digit hexadecimal number.
`controller-port-wwpn`	String (16)	Required	The WWPN of the storage controller port. The string form of a 16-digit hexadecimal number.
`lun`	String (16)	Required	The Logical Unit Number (LUN) of the storage resource. The string form of a 16-digit hexadecimal number.

A path-eckd object contains information about a single path by which an ECKD storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is **"eckd"**. A path-fcp object contains information about a single path by which an FCP storage resource is accessible to a virtualization host. This object describes a path for a virtualization host storage resource whose **type** property is either **"fcp"** or **"zvm-fcp"**.

Name	Type	Rqd/Opt	Description
`device-number`	String (1-4)	Required	The device number of the storage resource. The string form of a 1-4 digit hexadecimal number.

Description

The **Remove Virtualization Host Storage Resource Paths** operation removes path definitions from the specified virtualization host storage resource. The **controller-port-wwpn**, **lun**, or **device-number**, as appropriate, along with the **host-port-wwpn**, must identify a configured path of the storage resource associated with this virtualization host storage resource. If that condition is not met, HTTP status code 400 (Bad Request) is returned.

The URI path must designate an existing virtualization host object, and the API user must have object-access permission to it. Furthermore, the request body must designate an existing virtualization host storage resource object. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Remove Storage Resources**

task; otherwise, status code 403 (Forbidden) is returned. The virtualization host is marked busy for the duration of this request. If it is already marked busy due to some other operation, then status code 409 (Conflict) is returned.

Upon success, HTTP status code 204 (No Content) is returned and no response body is provided. If this operation changes the value of the **path** property, a property-change notification is issued asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	142	The specified path information does not identify a configured path available for the virtualization host to use to access the specified storage resource.
403 (Forbidden)	1	API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{virt-host-id}*) does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported.
	147	The URI in the request body (*{virtualization-host-storage-resource}*) does not designate an existing virtualization host storage resource object for the specified virtualization host.
409 (Conflict)	2	The operation cannot be performed because the virtualization host designated by the request URI is currently busy performing some other operation.
	150	The operation cannot be performed because the virtualization host designated by the request URI is currently busy due to a zBX Move operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	2	The request could not be processed because the HMC is not currently communicating with an element of a zBX needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- The **Remove Virtualization Host Storage Resource Paths** operation performs a portion of the function (remove path) provided by the Details for Storage Resource subtask of the **Manage Storage Resources** task. The operation performs some or all of the functions provided by the following Web Services APIs:
  - **Remove Virtualization Host Storage Resource Path**

    Removes a path between a virtualization host and a storage resource.
  - Gives a warning/error message in case an incorrect path or no path is selected.

Discover Virtualization Host Storage Resources

The **Discover Virtualization Host Storage Resources** operation discovers the virtualization host storage resources for a virtualization host.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/operations/discover-virtualization-host-storage-resources

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host object.

Query parameters

Name	Type	Rqd/Opt	Description
`prefix`	String (1-50)	Optional	Prefix to use when constructing the value of the **name** property of each newly discovered virtualization host storage resource. It must consist only of alphanumeric characters and the following special characters: "._-", and it must begin with an alphanumeric character.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`virtualization-host-storage-resources`	Array of objects	On successful completion, the response body contains a JSON object that provides an array whose elements are discovered-virtualization-host-storage-resource objects (described in the next table) including path accessibility status. Field names and data types in the discovered-virtualization-host-storage-resource are the same as the property names and data types defined in the data model. If no virtualization host storage resources are to be returned, an empty array is provided.

Each discovered-virtualization-host-storage-resource object contains the following fields:

Field name	Type	Description
`parent`	String/URI	The **parent** property of the virtualization host storage resource object.
`class`	String	The **class** property of the virtualization host storage resource object.
`name`	String	The **name** property of the virtualization host storage resource object.
`size`	Long	The **size** property of the virtualization host storage resource object.
`allocation-units`	String Enum	The **allocation-units** property of the virtualization host storage resource object.
`type`	String Enum	The **type** property of the virtualization host storage resource object.

Field name	Type	Description
storage-resource	String/URI	The **storage-resource** property of the virtualization host storage resource object. This will be null unless this virtualization host storage resource defines an additional path for the virtualization host to access an existing storage resource.
paths	Array of objects	The **paths** property of the virtualization host storage resource object.
unique-device-id	String	The **unique-device-id** property of the virtualization host storage resource object.
volume-serial-number	String (1-6)	The volume serial for this virtualization host storage resource. Only present if the **type** property is **"eckd"** or **"zvm-fcp"**.
device-number	String (1-4)	The device number that the virtualization host uses to access an ECKD storage resource, or the EDEV (emulated volume) that is created for FCP storage resources when they get allocated to a Virtual Server, or added to a Virtualization Host Storage Group. The string form of a 1-4 digit hexadecimal number. Only present if the **type** property is **"eckd"** or **"zvm-fcp"**.

Description

The **Discover Virtualization Host Storage Resources** operation discovers a virtualization host's virtualization host storage resources. The current values of the properties for the discovered virtualization host storage resource objects, including path accessibility status, are returned as defined in the"Data model" on page 314.

If the **prefix** parameter is omitted, the ensemble's default prefix will be used. The default prefix consists of the name of the ensemble concatenated with "_SR".

If there are no discovered virtualization host storage resources for the specified virtualization host, an empty array is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Discover Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 336.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because the virtualization host designated by the request URI is currently busy performing some other operation.
	150	The operation cannot be performed because the virtualization host designated by the request URI is currently busy due to a zBX Move operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Notifications

Changes to properties of a virtualization host storage resource are reflected via a property change notification designating the virtualization host as the managed object and the virtualization host storage resource as the element object. The standard property change notification fields (old value, new value and property name) for each changed property are provided in the notification.

The creation or deletion of a virtualization host storage resource is reflected via an inventory change notification designating the virtualization host as the managed object and the virtualization host storage resource as the element object.

Inventory service data

Information about the storage resources associated with a virtualization host can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for Virtualization Host Storage Resource objects are included in the response to the Inventory Service's Get Inventory operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"virtualization-host"** are to be included. An entry for a particular virtualization host storage resource is included only if the API user has object-access permission to the owning virtualization host.

Virtualization host storage group object

A virtualization host storage group is a representation of a z/VM Storage Group. It consists of homogeneous storage resources to which a z/VM virtualization host has access, and whose corresponding virtualization host storage resources have been placed into a storage group.

Data model

This object includes the following properties:

Table 64. Virtualization host storage group object properties

Name	Qualifier	Type	Description
element-uri	—	String/URI	Canonical URI path of the virtualization host storage group object, in the form `/api/virtualization-hosts/{virt-host-id}/virtualization-host-storage-groups/{virt-host-storage-group-id}` where *{virt-host-storage-group-id}* is the value of the **element-id** property of the virtualization host storage group.
element-id	—	String (36)	The unique identifier for the virtualization host storage group instance. This identifier is in the form of a UUID.
parent	—	String/URI	The parent object of a virtualization host storage group object is a virtualization host object.

Table 64. Virtualization host storage group object properties (continued)

Name	Qualifier	Type	Description
class	—	String	The class of a virtualization host storage group object is **"virtualization-host-storage-group"**.
name	—	String (1-64)	The name of the virtualization host storage group. It must consist only of alphanumeric characters, spaces and the following special characters: "._-$", and it must begin with an alphanumeric character or "$".
description	—	String (0-256)	The description for the virtualization host storage group. It must consist only of alphanumeric characters, spaces and the following special characters: "._-$".
free-space	—	Array of object	Information about the free space in the virtualization host storage group. It is an array of free-space-information objects, each of which describes an area of free storage in the virtualization host storage group. If there is no free space in the virtualization host storage group, an empty array is provided.
virtualization-host	—	String/URI	Canonical URI path of the virtualization host that owns this virtualization host storage group.

A free-space-information object contains information about an area of storage in the virtualization host storage group that is currently not in use.

Table 65. Virtualization host storage group object: free-space-information object properties

Name	Type	Description
device-number	String (1-4)	The device number of the virtualization host storage resource which has free space on it. This is the string form of a 1-4 digit hexadecimal number.
size	Long	The size of the free storage area. The units for this property are specified by the **allocation-units** property.
allocation-units	String Enum	The units for the size property. Values: • **"bytes"** • **"cylinders"**

Operations

If a virtualization host storage group operation accesses a z/VM virtualization host and encounters an error while communicating with the virtualization host via SMAPI, the response body is as described in "SMAPI Error Response Body" on page 203.

List Virtualization Host Storage Groups

The **List Virtualization Host Storage Groups** operation lists the virtualization host storage groups for a virtualization host.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-groups

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host object.

Query parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property.

Name	Type	Rqd/Opt	Description
properties	String	Optional	Identifies the properties of each virtualization host storage group to be returned. The only supported value is **"all"**, which results in all properties being returned. If this query parameter is omitted, a set of basic properties considered to be of general interest is returned.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
virtualization-host-storage-groups	Array of objects	If the **properties=all** query parameter is specified, an array is provided whose elements are the set of virtualization host storage group properties that would be returned on a **Get Virtualization Host Storage Group Properties** operation. If the **properties** query parameter is omitted, an array of virtualization-host-storage-group-basic-info objects is returned, described in the next table. If no virtualization host storage groups are to be returned, an empty array is provided.

Each virtualization-host-storage-group-basic-info object contains the following fields:

Field name	Type	Description
element-uri	String/URI	Canonical URI path of the virtualization host storage group object
name	String	The **name** property of the virtualization host storage group object

Description

The **List Virtualization Host Storage Groups** operation lists a virtualization host's virtualization host storage groups. The element URI and name are provided for each virtualization host storage group.

If the **name** query parameter is specified, the returned list is limited to those virtualization host storage groups that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not performed.

If the **properties** query parameter is specified, it controls the set of properties returned. A value of **"all"** results in all properties being returned, in exactly the same format as would be provided on a **Get Virtualization Host Storage Group Properties** operation. If the **properties** query parameter is omitted, a set of basic properties is returned for each virtualization host storage group. See the virtualization-host-storage-group-basic-info object definition. Any value other than **"all"** is not valid and results in an HTTP status code 400 (Bad Request).

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

If the virtualization host has no virtualization host storage groups, an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI

- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 340.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	145	A value other than **"all"** was specified for the **properties** query parameter, or this query parameter was specified more than once.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/virtualization-host-storage-groups HTTP/1.1
x-api-session: 1rmnds0imna61i3ll0eu7drk7jsec93mvclfbuqdb7xspk2fm5
```

Figure 168. List Virtualization Host Storage Groups: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Thu, 04 Aug 2011 13:30:20 GMT
content-type: application/json;charset=UTF-8
content-length: 683
{
   "virtualization-host-storage-groups": [
      {
         "element-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/
        virtualization-host-storage-groups/92560990-6aa4-11e0-b3b9-f0def10c03f4",
         "name": "$3390$"
      },
      {
         "element-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/
        virtualization-host-storage-groups/92559f6e-6aa4-11e0-b3b9-f0def10c03f4",
         "name": "$3380$"
      },
      {
         "element-uri": "/api/virtualization-hosts/342d80e0-65ff-11e0-acfd-f0def10c03f4/
        virtualization-host-storage-groups/925337e2-6aa4-11e0-b3b9-f0def10c03f4",
         "name": "$$FCP$$"
      }
   ]
}
```

Figure 169. List Virtualization Host Storage Groups: Response

Get Virtualization Host Storage Group Properties

The **Get Virtualization Host Storage Group Properties** operation retrieves the properties of a single virtualization host storage group object that is designated by its object ID and the object ID of the owning virtualization host.

HTTP method and URI

GET /api/virtualization-hosts/{*virt-host-id*}/virtualization-host-storage-groups/
{*virt-host-storage-group-id*}

URI variables

Variable	Description
{virt-host-id}	Object ID of the virtualization host
{virt-host-storage-group-id}	Element ID of the virtualization host storage group object for which properties are to be obtained

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the virtualization host storage group object as defined in "Data model" on page 338. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get Virtualization Host Storage Group Properties** operation returns the current properties for the virtualization host storage group object specified by *{virt-host-storage-group-id}* for the virtualization host specified by *{virt-host-id}*.

On successful execution, all of the current properties as defined by the data model for the virtualization host storage group object are provided in the response body, and HTTP status code 200 (OK) is returned.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. Furthermore, the URI path must designate an existing virtualization host storage group object. If any of these conditions are not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** task; otherwise, status code 403 (Forbidden) is returned.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 342.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID in the URI (*{virt-host-id}*) does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported. The object ID in the URI (*{virt-host-storage-group-id}*) does not designate an existing virtualization host storage group object for the specified virtualization host.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3/
  virtualization-host-storage-groups/305cf1fe-a8cf-11e0-9a45-00262df332b3 HTTP/1.1
x-api-session: 3gcd77glemvwq81dlmwxc8i4fwm4udxlby6i2auls4r6g529p1
```

Figure 170. Get Virtualization Host Storage Group Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Tue, 06 Dec 2011 14:34:05 GMT
content-type: application/json;charset=UTF-8
content-length: 466
{
   "class": "virtualization-host-storage-group",
   "description": "$$FCP$$",
   "element-id": "305cf1fe-a8cf-11e0-9a45-00262df332b3",
   "element-uri": "/api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3/
    virtualization-host-storage-groups/305cf1fe-a8cf-11e0-9a45-00262df332b3",
   "free-space": [],
   "name": "$$FCP$$",
   "parent": "/api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3",
   "virtualization-host": "/api/virtualization-hosts/0e4a5d94-a8c1-11e0-9492-00262df332b3"
}
```

Figure 171. Get Virtualization Host Storage Group Properties: Response

List Virtualization Host Storage Resources in a Virtualization Host Storage Group

The **List Virtualization Host Storage Resources in a Virtualization Host Storage Group** operation lists the virtualization host storage resources that are members of the specified virtualization host storage group.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/operations/list-virtualization-host-storage-resources-in-group

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host that owns the virtualization host storage group.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-host-storage-group`	String/URI	Optional	Canonical URI path of the virtualization host storage group whose members are to be listed. If none is specified, the operation returns information for all virtualization host storage groups.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`virtualization-host-storage-resources`	Array of objects	Array of virtualization-host-storage-resource-basic-info objects is returned, described in the next table. If no virtualization host storage resources are to be returned, an empty array is provided.

Each virtualization-host-storage-resource-basic-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the virtualization host storage resource object
`name`	String	The **name** property of the associated storage resource object
`type`	String Enum	The **type** property of the associated storage resource object

Description

The **List Virtualization Host Storage Resources in a Virtualization Host Storage Group** operation returns basic information about each virtualization host storage resource in the virtualization host storage group specified by {*virtualization-host-storage-group*} for the virtualization host specified by {*virt-host-id*}. If {*virtualization-host-storage-group*} is omitted, information from all of the virtualization host's virtualization host storage group is returned. The object URI and other basic properties are provided for each virtualization host storage resource.

A set of basic properties is returned for each virtualization host storage resource. See the virtualization-host-storage-resource-basic-info object definition.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If any of these conditions is not met, status code 404 (Not Found) is returned. If a URI is specified in the request body and it does not identify a virtualization host storage group for the specified virtualization host, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Manage Storage Resources** action; otherwise, status code 403 (Forbidden) is returned.

If there are no virtualization host storage resources in the target virtualization host storage group(s), an empty list is provided and the operation completes successfully.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Manage Storage Resources** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission to this operation.
404 (Not Found)	1	The object ID in the URI (*{virt-host-id}*) does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported.
	2	The URI specified in the request body does not identify a virtualization host storage group for the specified virtualization host.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Add Virtualization Host Storage Resource to Virtualization Host Storage Group

The **Add Virtualization Host Storage Resource to Virtualization Host Storage Group** operation adds a specified virtualization host storage resource to the appropriate virtualization host storage group.

HTTP method and URI

POST /api/virtualization-hosts/{*virt-host-id*}/operations/add-virtualization-host-storage-resource-to-group

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host that owns the virtualization host storage group.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-host-storage-resource`	String/URI	Optional	Canonical URI path of the virtualization host storage resource object to be added

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the virtualization host storage group to which the virtualization host storage resource was added in the form /api/virtualization-hosts/*{virt-host-id}*/ virtualization-host-storage-groups/*{virt-host-storage-group-id}*.

Field name	Type	Description
element-id	String	Element ID of the virtualization host storage group object. This is the *{virt-host-storage-group-id}* portion of the URI path provided by the **element-uri** field.

Description

The **Add Virtualization Host Storage Resource to Virtualization Host Storage Group** operation adds a specified virtualization host storage resource to the appropriate virtualization host storage group for the virtualization host specified by *{virt-host-id}*.

On successful execution, the virtualization host storage resource identified in "Request body contents" on page 346 has been added to the appropriate virtualization host storage group, and HTTP status code 200 (OK) is returned. "Response body contents" on page 346 identifies the virtualization host storage group to which the virtualization host storage resource was added.

On successful execution, a property-change notification is issued asynchronously to this operation. See "Notifications" on page 349 for more information.

The URI path must designate an existing virtualization host object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned. If the URI in the request body does not identify a virtualization host storage resource for the specified virtualization host, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Add Storage Resource to Group** task; otherwise, status code 403 (Forbidden) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Add Storage Resource to Group** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported.
	147	The URI in the request body (*{virtualization-host-storage-resource}*) does not designate an existing virtualization host storage resource object for the specified virtualization host.

HTTP error status code	Reason code	Description
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Remove Virtualization Host Storage Resource from Virtualization Host Storage Group

The **Remove Virtualization Host Storage Resource from Virtualization Host Storage Group** operation removes a specified virtualization host storage resource from the appropriate virtualization host storage group.

HTTP method and URI

```
POST /api/virtualization-hosts/{virt-host-id}/operations/remove-virtualization-host-
  storage-resource-from-group
```

In this request, the URI variable *{virt-host-id}* is the object ID of the virtualization host that owns the virtualization host storage resource.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`virtualization-host-storage-resource`	String/URI	Required	Canonical URI path of the virtualization host storage resource object to be removed

Description

The **Remove Virtualization Host Storage Resource from Virtualization Host Storage Group** operation removes a specified virtualization host storage resource from the appropriate virtualization host storage group for the virtualization host specified by *{virt-host-id}*.

On successful execution, the virtualization host storage resource identified in "Request body contents" has been removed from the appropriate virtualization host storage group, and HTTP status code 204 (No Content) is returned and no response body is provided.

On successful execution, a property-change notification is issued asynchronously to this operation. See "Notifications" on page 349 for more information.

The URI path must designate an existing virtualization host object, and the API user must have object-access permission to it. Furthermore, the request body must designate an existing virtualization host storage resource object for the specified virtualization host. If any of these conditions is not met, status code 404 (Not Found) is returned. In addition, the API user must have action access permission to the **Remove Storage Resource from Group** task; otherwise, status code 403 (Forbidden) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the hosting object of the virtualization host specified in the request URI
- Action/task permission to the **Remove Storage Resource from Group** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	API user does not have action permission to this operation.
404 (Not Found)	1	The object ID in the URI *{virt-host-id}* does not designate an existing virtualization host object, or the API user does not have object access permission to it, or it is not of a type for which this operation is supported.
	147	The URI in the request body (*{virtualization-host-storage-resource}*) does not designate an existing virtualization host storage resource object for the specified virtualization host.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not currently communicating with an SE needed to perform the requested operation.
	100	The request could not be processed because the HMC is unable to communicate with a z/VM virtualization host via SMAPI.
	101	The request could not be processed because a command executed on a z/VM virtualization host via SMAPI failed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Notifications

Changes to properties of a virtualization host storage group are reflected via a property change notification designating the virtualization host as the managed object and the virtualization host storage group as the element object. The standard property change notification fields (old value, new value and property name) for each changed property are provided in the notification.

The addition or removal of a virtualization host storage resource to/from a virtualization host storage group is reflected via a property change notification designating the virtualization host as the managed object and the virtualization host storage group as the element object. The changed property identified in the notification is the **virtualization-host-storage-resources** property. Note that this is not included as a property of a virtualization host or a virtualization host storage group in their respective data models; this property name is only used in the context of these property change notifications. When the notification is due to the addition of a virtualization host storage resource, the new value in the notification is the URI path of the added virtualization host storage resource and the old value is null. When the notification is due to the removal of a virtualization host storage resource, the old value in the notification is the URI path of the removed virtualization host storage resource and the new value is **null**. Thus, unlike other property change notifications, these notifications only identify the delta to the set of

members of the virtualization host storage group rather than the complete new and old membership sets. In that sense, they are more like inventory change notifications.

Inventory service data

Information about the storage groups associated with a virtualization host can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for Virtualization Host Storage Group objects are included in the response to the Inventory Service's **Get Inventory** operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"virtualization-host"** are to be included. An entry for a particular virtualization host storage group is included only if the API user has object-access permission to the owning virtualization host.

Usage notes

- The group to which a virtualization host storage resource is added is determined by the type of the resource. Specifically, FCP resources are added to a group named “$$FCP$$”, 3380 resources are added to the “3380” group, and 3390 resources are added to the “3390” group. The user may not specify any attributes of these groups (e.g., name or description). Thus, there are no writeable properties of a group.
- It is not possible for a virtual disk to span multiple physical storage devices. Thus, the largest virtual disk that may be created in the virtualization host storage group is limited to the largest free space on any single virtualization host storage resource in the group. The information describing all such areas of free space may be useful when defining, or planning to define, multiple virtual disks in the group.

Chapter 12. Virtual network management

In a zEnterprise ensemble, the Intra-Ensemble Data Network (IEDN) provides the physical, layer 2 network to which all System z CECs and the blades in the zBX are attached. The zManager Manage Virtual Network tasks are used to provision the IEDN into virtual networks, and to manage those virtual networks. In an ensemble, a virtual network is defined by a name, VLAN ID, and zero or more associated network host interfaces of the following types: virtual server, optimizers, and Top-of-Rack (TOR) switch ports that attach ISAOPT and external routers. The ensemble has a default virtual network. This virtual network cannot be deleted, although its VLAN ID can be changed.

To communicate on the IEDN, a network host's network interfaces must be associated with a virtual network that is managed by zManager. Note that the different types of network hosts have different requirements for defining and configuring network interfaces. For example, part of the process of defining a virtual server is to create its virtual network interfaces and associate them with at virtual network. The same is true for most optimizers. In the case of ISAOPT, the association to a virtual network is a special case. For ISAOPT, the TOR port that attaches to the coordinator blade's access switch provides the virtual network interface attachment. In cases where the TOR's external ports are attached to external networking equipment, such as a router, these TOR ports are required to join one or more virtual networks defined by zManager.

Virtual network management operations summary

The following tables provide an overview of the operations provided.

Table 66. Virtual network management: operations summary

Operation name	HTTP method and URI
"List Virtual Networks" on page 352	`GET /api/ensembles/{ensemble-id}/virtual-networks`
"Get Virtual Network Properties" on page 354	`GET /api/virtual-networks/{virtual-network-id}`
"Update Virtual Network Properties" on page 355	`POST /api/virtual-networks/{virtual-network-id}`
"Create Virtual Network" on page 358	`POST /api/ensembles/{ensemble-id}/virtual-networks`
"Delete Virtual Network" on page 360	`DELETE /api/virtual-networks/{virtual-network-id}`
"List Members of a Virtual Network" on page 362	`GET /api/virtual-networks/{virtual-network-id}/host-vnics`

Table 67. Virtual network management: URI variables

Variable	Description
{ensemble-id}	Object ID of an Ensemble object
{virtual-network-id}	Object ID of a Virtual Network

Virtual network object

A virtual network object represents a single zEnterprise virtual network.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, with the following type-specific specialization:

Table 68. Virtual network object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/URI	The canonical URI path for a virtual network object is of the form `/api/virtual-networks/{virtual-network-id}` where *{virtual-network-id}* is the value of the **object-id** property of the virtual network object.
parent	—	String/URI	The parent of a virtual network object is an ensemble object.
class	—	String	The class of a virtual network object is **"virtual-network"**.
name	(w)(pc)	String (1-32)	The name of the virtual network object as known by zManager. This name must be unique across virtual networks. If the virtual network is the default virtual network, then the name cannot be changed. The name provided must be between 1 and 32 characters.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional type-specific properties:

Table 69. Virtual network object: class specific additional properties

Name	Qualifier	Type	Description
vlan-id	(w)(pc)	Integer	The VLAN ID assigned to this virtual network. Valid VLAN IDs are in the range of 10-1030.
is-default	—	Boolean	This value is true when the virtual network is the default virtual network.
has-members	(pc)	Boolean	This value is true when the virtual network has member network host interfaces attached to it.

List Virtual Networks

Use the **List Virtual Networks** operation lists the virtual networks managed by the HMC.

HTTP method and URI

`GET /api/ensembles/{ensemble-id}/virtual-networks`

In this request, the URI variable *{ensemble-id}* is the object ID of an Ensemble object.

Query parameters

Name	Type	Rqd/Opt	Description
`is-default`	Boolean	Optional	Filter pattern to limit returned objects to those that have a matching is-default property. There is only one default virtual network, and it is always defined, so when is-default=true is specified, the response will be an array with one element.
`name`	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching name property. If a match is found, the response will be an array with one element. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`virtual-networks`	Array of obects	Array of nested virtual-network-info objects as described in the next table.

Each nested virtual-network-info object contains the following:

Field name	Type	Description
`object-uri`	String/ URI	Canonical URI path of the virtual network object, in the form `/api/virtual-networks/{virtual-network-id}`.
`name`	String	Display name of the virtual network object.

Description

This operation lists the virtual networks that are in the specified ensemble. The response body content is an array of one or more URI's that represent each virtual network. Each ensemble always has a single default virtual network with the **name** of **"Default"**.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the target Ensemble object
- Action/task permission to the **Manage Virtual Networks** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	1	The request included an unrecognized or unsupported query parameter.
403 (Forbidden)	1	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the Manage Virtual Networks task is required.
404 (Not Found)	1	The request URI does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/virtual-networks HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
```

Figure 172. List Virtual Networks: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 05:11:15 GMT
content-type: application/json;charset=UTF-8
content-length: 316
{
   "virtual-networks": [
      {
         "name": "Default",
         "object-uri": "/api/virtual-networks/f920171e-03f2-11e1-8e8e-0010184c8334"
      },
      {
         "name": "SS-Web-Store-Network",
         "object-uri": "/api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334"
      }
   ]
}
```

Figure 173. List Virtual Networks: Response

Get Virtual Network Properties

Use the **Get Virtual Network Properties** operation retrieves the properties of a single Virtual Network object that is designated by the *{virtual-network-id}*.

HTTP method and URI

GET /api/virtual-networks/{*virtual-network-id*}

In this request, the URI variable *{virtual-network-id}* is the object ID of the Virtual Network object for which properties are to be obtained.

Response body contents

On successful completion, the response body contains a JSON object that provides the current values of the properties for the virtual network object as defined in the Data Model section above. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

TThis operation returns the current properties for the Virtual Network object. In an ensemble, there is at least one virtual network. This is the default virtual network with the **name "Default"**.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the target Virtual Network
- Action/task permission to the **Manage Virtual Networks** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 354.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Manage Virtual Networks** task is required.
404 (Not Found)	1	The request URI does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334 HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
```

Figure 174. Get Virtual Network Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 05:11:15 GMT
content-type: application/json;charset=UTF-8
content-length: 368
{
   "class": "virtual-network",
   "description": "Spacely Sprockets Web Store Network",
   "has-members": true,
   "is-default": false,
   "is-locked": false,
   "name": "SS-Web-Store-Network",
   "object-id": "e58564e0-1723-11e1-aea4-0010184c8334",
   "object-uri": "/api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334",
   "parent": "/api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334",
   "vlan-id": 1030
}
```

Figure 175. Get Virtual Network Properties: Response

Update Virtual Network Properties

Use the **Update Virtual Network Properties** operation updates one or more of the writeable properties of a Virtual Network object.

HTTP method and URI

POST /api/virtual-networks/{*virtual-network-id*}

In this request, the URI variable *{virtual-network-id}* is the object ID of the Virtual Network object for which properties are to be updated.

Request body contents

The request body is expected to contain a JSON object that provides the new values of any writeable property that is to be updated by this operation. Field names and data types in this JSON object are expected to match the corresponding property names and data types defined by the data model for this object type. The JSON object can and should omit fields for properties whose values are not to be changed by this operation.

Description

This operation updates writeable properties of the virtual network object specified by *{virtual-network-id}*.

If the **vlan-id** property is changed, and there are member attached hosts are not in the proper operating status to accept a virtual network change, then this request will fail with HTTP status code 409, and the response body will contain the list of URI's of the network hosts that were unable to accept a virtual network change. Upon failure, the virtual network's vlan-id is not changed, although requests for updates to other virtual network properties may have successfully occurred. Upon successful complete, the virtual network of the attached network hosts virtual network interfaces will be changed.

The request body is validated against the schema described in "Request body contents." If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

The request body does not need to specify a value for all writeable properties, but rather can and should contain fields for the properties to be updated. Object properties for which no input value is provided remain unchanged by this operation.

If the update changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the target Virtual Network object
- Action/task permission to the **Virtual Networks Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	129	The default virtual network name of **"Default"** cannot be changed.
	131	The specified name or VLAN ID for a virtual network already exists. This includes specification of the name **"Default"**.
403 (Forbidden)	1	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Manage Virtual Networks** task is required.
404 (Not Found)	1	The object-id in the URI *{virtual-network-id}* does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.
409 (Conflict)	130	One or more of the network attached hosts that are members of this virtual network do not have the proper operating status to support changes to the virtual network at this time. The URIs of network hosts that were unable to accept the virtual network change are returned in the response body. Retry the request. Currently PowerVM and x Hyp servers have operating status restrictions on changes to a virtual network. See the network-adapter Data Model for details for virtual server status that allows network-adapter changes.
503 (Service Unavailable	1	Communication between the HMC and SE is unavailable. Please retry the request.
	2	The request could not be processed because the SE is not currently communicating with an element of a zBX needed to perform the requested operation.

On completion where the HTTP status code is 409 (Conflict), the standard error response body contains an error-details field that provides a list of network hosts that were unable to accept a virtual network change. The value of the error-details field is a nested object with the following fields:

Field name	Type	Description
hosts-list	Array of string/URI	Array of URIs that identify the network hosts that were unable to accept a virtual network change. The URIs for the network hosts are in the following forms: Virtual server: `/api/virtual-servers/{virtual-server-id}` Optimizer: `/api/blades/{blade-id}`

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334 HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
content-type: application/json
content-length: 103
{
   "description": "Spacely Sprockets Web Store Network",
   "name": "SS-Web-Store-Network",
   "vlan-id": 1030
}
```

Figure 176. Update Virtual Network Properties: Request

```
204 No Content
date: Fri, 25 Nov 2011 05:11:13 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 177. Update Virtual Network Properties: Response

Create Virtual Network

Use the **Create Virtual Network** operation creates a new virtual network in the ensemble.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/virtual-networks

In this request, the URI variable *{ensemble-id}* is the object ID of the Ensemble object to which the Virtual Network is to be added.

Request body contents

The request body is expected to contain a JSON object with writeable network properties of the virtual network object that will be used to create the virtual network:

Field name	Type	Rqd/Opt	Description
name	String	Required	The name of the virtual network object as known by zManager. This name must be unique across all virtual networks, and cannot be the value **"Default"** as this name is reserved as the zManager-defined name of the default virtual network. The name provided must be between 1 and 32 characters.
description	String	Optional	Display description of the Virtual Network object.
vlan-id	Integer	Required	The VLAN ID assigned to this virtual network. Valid VLAN IDs are in the range of 10-1030. This value must be unique across all virtual networks.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the Virtual Network object, in the form `/api/virtual-networks/{virtual-network-id}`.

Description

This operation creates a new virtual network in the ensemble specified by *{ensemble-id}*. A virtual network with the same **vlan-id** or **name** must not already exist in the ensemble.

On successful execution of this operation the virtual network is created using the inputs as specified by the input fields of the request body. The URI of the new virtual network is provided in the response body and in a **Location** response header as well.

The request body is validated against the schema described in the Request Body Contents section. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the target Ensemble object
- Action/task permission to the **New Virtual Networks** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	0	This error indicates maximum number of virtual networks exceeded or unavailable resources.
	131	The specified name or VLAN ID for a virtual network already exists. This includes specification of the name **"Default"**.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform the requested action.
404 (Not Found)	1	The object-id in the URI *{ensemble-id}* does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/virtual-networks HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
content-type: application/json
content-length: 71
{
   "description": "New Network",
   "name": "SS-New-Network",
   "vlan-id": 11
}
```

Figure 178. Create Virtual Network: Request

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334
cache-control: no-cache
date: Fri, 25 Nov 2011 05:11:13 GMT
content-type: application/json;charset=UTF-8
content-length: 75
{
   "object-uri": "/api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334"
}
```

Figure 179. Create Virtual Network: Response

Delete Virtual Network

Use the **Delete Virtual Network** operation removes a virtual network from an ensemble.

HTTP method and URI

DELETE /api/virtual-networks/{*virtual-network-id*}

In this request, the URI variable *{virtual-network-id}* is the object ID of the Virtual Network object to be deleted.

Description

On successful completion, this operation removes the virtual network specified by the URI from the ensemble. Any network attached hosts that are members of this virtual network and have the proper operating status to accept a virtual network change will be removed from the virtual network. Currently only PowerVM and x Hyp servers require the proper operating status to accept a virtual network change. If attached hosts cannot be removed from the virtual network, then this request will fail with an HTTP status code of 409, and the response body will contain the list of the URIs of the network hosts that were unable to accept a virtual network change. Upon failure, no network attached hosts are removed from the virtual network, and the virtual network is not deleted.

Note that the default virtual network, **"Default"**, cannot be deleted from the ensemble.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the target Ensemble object
- Action/task permission to the **New Virtual Networks** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	132	The default virtual network cannot be deleted.
404 (Not Found)	1	The object-id in the URI does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.
409 (Conflict)	130	One or more of the network attached hosts that are members of this virtual network do not have the proper operating status to support a change to the virtual network at this time. The URIs of network hosts that were unable to accept the virtual network change are returned in the response body. Retry the request. Currently PowerVM and x Hyp type servers have operating status restrictions on changes to a virtual network. See the network-adapter Data Model for details for virtual server status that allows network-adapter changes.
503 (Service Unavailable)	1	Communication between the HMC and SE is unavailable. Please retry the request.
	2	The request could not be processed because the SE is not currently communicating with an element of a zBX needed to perform the requested operation.

On completion where the HTTP status code is 409 (Conflict), the standard error response body contains an error-details field that provides a list of network hosts that were unable to accept a virtual network change. The value of the error-details field is a nested object with the following fields:

Field name	Type	Description
hosts-list	Array of string/URI	Array of URIs that identify the network hosts that were unable to accept a virtual network change. The URIs for the network hosts are in the following forms: Virtual server: `/api/virtual-servers/{`*`virtual-server-id`*`}` Optimizer: `/api/blades/{`*`blade-id`*`}`

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334 HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
```

Figure 180. Delete Virtual Network: Request

```
204 No Content
date: Fri, 25 Nov 2011 05:11:15 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 181. Delete Virtual Network: Response

List Members of a Virtual Network

Use the **List Members of a Virtual Network** operation lists the members of a Virtual Network.

HTTP method and URI

GET/api/virtual-networks/{*virtual-network-id*}/host-vnics

In this request, the URI variable *{virtual-network-id}* is the object ID of the Virtual Network for which members are to be listed.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
host-vnics	Array of objects	Array of nested member-info objects, each element of which identifies an attached network host's network interface that is a member of this virtual network. The nested member-info object is described in the next table. If the virtual network has no network interfaces associated with it then an empty array is returned.

Each nested member-info object contains the following:

Field name	Type	Description
object-uri	String/URI	A URI of an object or element that identifies an attached network hosts' network interface that is a members of this virtual network. A network interface can be one of the following: A network adapter of a virtual server or optimizer, or a Top-of-Rack-Switch (TOR) port that is attached to an ISAOPT optimizer or to an external router. The URI's for the network hosts/interfaces are: Virtual server: /api/virtual-servers/{*virtual-server-id*}/network-adapters/{*network-adapter-id*} Optimizer: /api/blades/{*blade-id*}/iedn-interfaces/{*iedn-interface-id*} TOR: /api/zbxs/{*zbx-id*}/top-of-rack-switches/{*tor-id*}/ports/{*port-id*}

Description

The List Members of a Virtual Network operation lists the members of a Virtual Network. The members of a virtual network are network attached hosts' network interfaces which include the following types: virtual servers' virtual network interfaces, optimizer virtual network interfaces, and Top-of-Rack (TOR) switch network interfaces (ports).

On successful execution, this operation returns an array of nested objects that identify the members of the Virtual Network specified by the request URI. This array may be empty if the virtual network has no associated network hosts. In some cases, a network host may have multiple interfaces associated with this virtual network.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the target Virtual Network object
- Object access to the specific member attached network host object
- Action/task permission to the **Manage Virtual Networks** task.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 362.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user under which the API request was authenticated is not authorized to perform the requested operation. Permission to the **Manage Virtual Networks** task is required.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The request URI does not designate an existing resource of the correct type, or designates a resource for which the API user does not have object-access permission.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
GET /api/virtual-networks/e58564e0-1723-11e1-aea4-0010184c8334/host-vnics HTTP/1.1
x-api-session: 5tjfpg1d6i7dfria4i52czubs5ptly4fqig2gm3mkluph6ebga
```

Figure 182. List Members of a Virtual Network: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 05:11:15 GMT
content-type: application/json;charset=UTF-8
content-length: 274
{
   "host-vnics": [
      {
         "object-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af/
       network-adapters/596dd87c-0db7-11e1-9251-f0def14b63af"
      },
      {
         "object-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af/
       network-adapters/582b5d0e-0db7-11e1-b1f1-f0def14b63af"
      }
   ]
}
```

Figure 183. List Members of a Virtual Network: Response

Inventory service data

Information about the Virtual Networks managed by the HMC can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for virtual network objects are included in the response to the Inventory Service's Get Inventory operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class **"virtual-network"** are to be included. An entry for a particular virtual network is included only if the API user has object-access permission to that object.

For each virtual network object to be included, the inventory response array includes entry that is a JSON object with the same contents as is specified in the Response Body Contents section for the Get Virtual Network Properties operation. That is, the data provided is the same as would be provided if a Get Virtual Network Properties operation were requested targeting this object.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the **Get Inventory** response to describe a single virtual network. This object would appear as one array entry in the response array:

```
{
   "class": "virtual-network",
   "description": "Network for servers of the Shimmer order processing application.",
   "has-members": false,
   "is-default": false,
   "name": "Shimmer Order Processing Network",
   "object-id": "e3a366f4-95e6-11e0-85c7-000c29bb873c",
   "object-uri": "/api/virtual-networks/e3a366f4-95e6-11e0-85c7-000c29bb873c",
   "parent": "/api/ensembles/890b6df2-93a4-11e0-887c-000c29bb873c",
   "vlan-id": 1008
}
```

Figure 184. Virtual network object: Sample inventory data

Chapter 13. Workload resource group management

This chapter describes workload resource group management APIs, and provides a data model and operations for each of the following: workload objects, performance policy objects, and performance management reports.

Overview

The IBM zEnterprise System (zEnterprise) extends performance management capability to both traditional System z and BladeCenter hardware environments. For multitier applications that span System z hardware and BladeCenter hardware, this extended capability enables dynamic adjustments to processor (CPU) allocations to ensure that those applications are provided with sufficient resources. To manage hardware resources in the zEnterprise environment, you group these resources into workload resource groups and define performance policies for them. zManager uses these policy definitions to manage the resources for each workload resource group in an ensemble.

In computing terminology, the usual definition of the term *workload* is the amount of application processing that a computer performs at a given time, and this processing usually includes a specific number of connected users interacting with the running applications. This amount of work often serves as a benchmark for computer performance.

In contrast, a zEnterprise workload resource group, which is sometimes referred to more simply as a "workload", is a customer-defined collection to be tracked, managed, and reported as a unit that reflects a business goal or function. This collection consists of virtualized hardware rather than software resources. Throughout the zEnterprise documentation, the term *workload resource group* means a collection of logical constructs called *virtual servers* that perform a customer-defined collective purpose.

Initially, after zEnterprise hardware is installed and the ensemble is configured, all virtual servers are associated with the default workload resource group and its default performance policy. Workload administrators use the HMC to create and name the new workload resource groups, and to define which virtual servers are associated with this workload resource group.

After a custom workload resource group is defined, an administrator can use ensemble tasks in the HMC to create performance policies that describe the workload resource group performance objectives and importance, and to create service classes that set priority for and classify work within a policy. These values govern performance management. Each workload resource group can have one or more performance policies that describe the performance objectives and importance. Each performance policy has service classes that set the priority of and classify resources within the policy. Only one performance policy within a specific workload resource group can be active at any given time; having multiple policies with different defined goals gives an administrator the ability to quickly change priorities by merely activating a different policy. zManager uses the active performance policy and its service classes to manage how physical resources are applied to the virtual servers associated with the workload resource group.

The Web Services API provides the same workload resource group management capabilities as those provided through HMC tasks, including:
- Creating, updating and deleting workload resource groups, performance policies, or service classes.
- Activating a performance policy.

In addition, zManager also continuously collects performance-management data that is available through a variety of reports that you view through the HMC. By setting a time interval for any report, you can query historical performance data as it applied over that time period. zManager keeps only the last 36 hours of collected performance data, so that limit defines the maximum duration for reports. Through

HMC tasks, these various reports present performance data in tables, charts or graphs. In many cases, you can request one report and select data within that report to request more detailed performance data that is available in another report type. For example, through the HMC you can select the **Workloads Report** task to list all workload resource groups defined in the ensemble. From that list, you can select a specific workload resource group to view a **Virtual Server Report** that lists all virtual servers defined to that specific workload resource group.

The Web Services API provide the same performance reports as those provided through HMC tasks, returning the same data and including the capability to query time intervals. With the zManager API, the caller requests a specific report through an operations call to generate and return the specific type of report. Data returned for items in one report can be used in requests for subsequent drill-down report generation requests for other report types.

For more information about workload resource group and performance management, including sample reports as they are returned through the HMC, see the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607.

Workload resource group operations summary

The following tables list the workload resource group operations that are provided through the Web Services API.

Table 70. Workload resource group management: operations summary

Operation name	HTTP method and URI path
"List Workload Resource Groups of an Ensemble" on page 373	`GET /api/ensembles/{ensemble-id}/workload-resource-groups`
"Get Workload Resource Group Properties" on page 375	`GET /api/workload-resource-groups/{workload-id}`
"Create Workload Resource Group" on page 377	`POST /api/ensembles/{ensemble-id}/workload-resource-groups`
"Delete Workload Resource Group" on page 380	`DELETE /api/workload-resource-groups/{workload-id}`
"Update Workload Resource Group" on page 381	`POST /api/workload-resource-groups/{workload-id}`
"List Virtual Servers of a Workload Resource Group" on page 383	`GET /api/workload-resource-groups/{workload-id}/virtual-servers`
"Add Virtual Server to a Workload Resource Group" on page 386	`POST /api/workload-resource-groups/{workload-id}/operations/add-virtual-server`
"Remove Virtual Server from a Workload Resource Group" on page 387	`POST /api/workload-resource-groups/{workload-id}/operations/remove-virtual-server`
"List Groups of Virtual Servers of a Workload Resource Group" on page 389	`GET /api/workload-resource-groups/{workload-id}/virtual-server groups`
"Add Group of Virtual Servers to a Workload Resource Group" on page 391	`POST /api/workload-resource-groups/{workload-id}/operations/add-virtual-server-group`
"Remove Group of Virtual Servers from a Workload Resource Group" on page 393	`POST /api/workload-resource-groups/{workload-id}/operations/remove-virtual-server-group`
"List Performance Policies" on page 400	`GET /api/workload-resource-groups/{workload-id}/performance-policies`

Table 70. Workload resource group management: operations summary (continued)

Operation name	HTTP method and URI path
"Get Performance Policy Properties" on page 402	`GET /api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}`
"Create Performance Policy" on page 405	`POST /api/workload-resource-groups/{workload-id}/performance-policies`
"Delete Performance Policy" on page 407	`DELETE /api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}`
"Update Performance Policy" on page 409	`POST /api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}`
"Activate Performance Policy" on page 412	`POST /api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}/operations/activate`
"Import Performance Policy" on page 413	`POST /api/workload-resource-groups/{workload-id}/operations/import-performance-policy`
"Export Performance Policy" on page 415	`POST /api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}/operations/export`
"Generate Workload Resource Groups Report" on page 419	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-workload-resource-groups-report`
"Generate Workload Resource Group Performance Index Report" on page 423	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-workload-resource-group-performance-index-report`
"Generate Workload Resource Group Resource Adjustments Report" on page 426	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-workload-resource-group-resource-adjustments-report`
"Generate Virtual Servers Report" on page 430	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-virtual-servers-report`
"Generate Virtual Server CPU Utilization Report" on page 434	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-virtual-server-cpu-utilization-report`
"Generate Virtual Server Resource Adjustments Report" on page 437	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-virtual-server-resource-adjustments-report`
"Generate Hypervisor Report" on page 442	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-hypervisor-report`
"Generate Hypervisor Resource Adjustments Report" on page 449	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-hypervisor-resource-adjustments-report`
"Generate Service Classes Report" on page 453	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-service-classes-report`
"Generate Service Class Resource Adjustments Report" on page 456	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-service-class-resource-adjustments-report`
"Generate Service Class Hops Report" on page 461	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-service-class-hops-report`
"Generate Service Class Virtual Server Topology Report" on page 466	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-service-class-virtual-server-topology-report`
"Generate Load Balancing Report" on page 474	`POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-load-balancing-report`
"Get Performance Management Velocity Level Range Mappings" on page 476	`GET /api/ensembles/{ensemble-id}/performance-management/velocity-level-range-mappings`

Table 71. Workload management: URI variables

Variable	Description
{ensemble-id}	Object ID (UUID) of an ensemble object
{workload-id}	Object ID (UUID) of a workload object
{policy-id}	ID (UUID) of a performance policy object within a workload resource group

Workload resource group object

A *workload object* is a managed object representing a group of virtual servers in a zEnterprise ensemble. Virtual servers can be managed in workload resource groups either by adding or removing them directly to or from the workload resource group, or through user-defined managed object groups. By adding a group to a workload resource group, all of the virtual servers in that group are implicitly added to the workload resource group. By adding or removing a virtual server to or from a group, that virtual server is implicitly added to or removed from the workload resource group. A virtual server can belong to more than one custom workload resource group. Any virtual server that does not belong to a custom workload resource group is placed in the default workload resource group.

A workload resource group contains a default performance policy as well as any custom performance policies created for it. Exactly one performance policy is active for a specific workload resource group at any given time, and is applied to the virtual servers in that workload resource group. The default workload resource group also has a default performance policy that is always active.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, with the type-specific specialization described in the following tables. Note that this object does not maintain the concept of an operational status, and therefore does not provide the operational-status-related properties.

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

Table 72. Workload object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/URI	The canonical URI path for a workload object is of the form `/api/workload-resource-groups/{workload-id}` where *{workload-id}* is the value of the **object-id** property of the workload object.
object-id	—	String (36)	The unique identifier for the workload resource group instance. This identifier is in the form of a UUID.
parent	—	String/URI	The parent of a workload resource group is conceptually its owning ensemble, so the parent value is the canonical URI path for the ensemble.
class	—	String	The class of a workload object is **"workload-resource-group"**.
name	(w) (pc)	String (1-64)	The display name specified for the workload resource group, which can be up to 64 characters made up of alphanumeric characters, blanks, periods, underscores, or dashes. Names must start with an alphabetic character and end with an alphabetic character, numeric character, underscore, period, or dash. Names must be unique to other existing workload resource groups in the ensemble.

Table 72. Workload object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
description	(w) (pc)	String (0-256)	Arbitrary text describing the workload resource group in up to 256 characters.
status	(sc)	String Enum	The current status of the workload resource group, which must be one of the following values: **"compliant"** The contributors to workload compliance are all currently compliant. **"not-compliant"** One or more of the contributors to the workload compliance are currently not-compliant.
acceptable-status	(w)(pc)	Array of String Enum	The set of status values that the workload resource group can be in and be considered to be in an acceptable (not alert causing) state. By default, this is **"compliant"** for workloads, and **"not-compliant", "compliant"** for the default workload.

In addition to the properties defined through included schemas, this object includes the additional type-specific properties in Table 73.

Table 73. Workload object: type-specific properties

Name	Qualifier	Type	Description of specialization
is-default	—	Boolean	This value identifies the default workload object. It is true for the default workload resource group and false for all other (custom) workload resource group in the ensemble.
category	(w) (pc)	String (0-32)	Up to 32 characters used to categorize the workload object, often with other workload resource groups within the ensemble.
virtual-servers	(c) (pc)	Array of String/URI	The complete list of all virtual servers in the workload, each identified by its URI, including those directly placed as well as those placed due to membership in a group. (This list corresponds to the list provided by the **List Virtual Servers of a Workload Resource Group** operation.) If the workload contains no virtual servers, then an empty array is provided. This virtual severs provided in this list can change as a result of the Add and Remove Virtual Server or the Add and Remove Groups operations on the workload. **Note:** This property is not returned by the **Get Workload Resource Group Properties** operation.
directly-added-virtual-servers	(c) (pc)	Array of String/URI	The list of virtual servers that have been directly placed in the workload, each identified by its URI. If the workload contains no directly placed virtual servers, then an empty array is provided. This value can be modified through the Add and Remove Virtual Server operations on the workload. **Note:** This property is not returned by the **Get Workload Resource Group Properties** operation.
virtual-server-groups	(c) (pc)	Array of String/URI	The list of the user-defined managed object groups, each identified by its URI, in the workload whose child virtual servers automatically become members of the workload. If the workload has no groups, an empty array is provided. This value can be modified through the Add and Remove Group operations on the workload. **Note:** This property is not returned by the **Get Workload Resource Group Properties** operation.

Table 73. Workload object: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
active-perf-policy	(pc)	Object	A perf-policy-summary object, as described in Table 74 on page 373, of the active performance policy in the workload resource group. This value can be modified through the operation "Activate Performance Policy" on page 412.
perf-activation-node-count	(pc)	Integer (0-n)	The number of nodes (ensemble members) that have successfully activated the active performance policy for the workload resource group. This value is between 0 and the total number of ensemble members.
perf-activation-status	(pc)	String Enum	The status of the last performance policy activation. The possible values are: **"initializing"** The workload resource group performance function is being initialized and status is not yet known. **"in-progress"** Performance policy activation is in progress and any new activation request is rejected. **"active"** Performance policy activation is complete.
default-perf-policy	—	Object	A perf-policy-summary object, as described in Table 74 on page 373, of the default performance policy in the workload resource group.
custom-perf-policies	(c) (pc)	Array of objects	A list of perf-policy-summary objects, as described in Table 74 on page 373, of the user-defined performance policies in the workload resource group. If no custom policies exist, an empty array is provided. This value can be modified through the operations "Create Performance Policy" on page 405 and "Delete Performance Policy" on page 407.
perf-status	(pc)	String Enum	The performance status of the workload, determined by the importance and PI value of the active performance policy's service classes. Possible values are: **"goals-met"** The PI values of all service classes in the active policy are less than or equal to one. **"exposed"** The PI value of a service class in the active policy is greater than one and its importance is **"low"** or **"lowest"**. **"severe"** The PI value of a service class in the active policy is greater than one and its importance is **"medium"**. **"critical"** The PI value of a service class in the active policy is greater than one and its importance is **"high"** or **"highest"**. **"no-status"** Performance status of the workload cannot be calculated.

Table 73. Workload object: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
compliant-perf-status	(w)(pc)	Array of String Enum	An array of values that define what **perf-status** values cause the workload's status to be compliant. Possible values are: **"goals-met"** The workload's performance status is **"goals-met"**. **"exposed"** The workload's performance status is **"exposed"**. **"severe"** The workload's performance status is **"severe"**. **"critical"** The workload's performance status is **"critical"**. **"no-status"** The workload's performance status is **"no-status"**. By default, a workload's compliant performance statuses include **"goals-met"**.

Table 74. perf-policy-summary-object

Field name	Type	Description
element-uri	String/URI	Canonical URI path of the performance policy object.
element-id	String	The **element-id** of the performance policy object.
name	String	Display name of the performance policy object.
activation-status	String Enum	The activation status of the performance policy object. The value is one of the following: **"not-active"** The performance policy is not currently the active policy in the workload resource group. **"in-progress"** The performance policy is currently being activated in the workload resource group. **"active"** The performance policy is currently the active policy in the workload resource group and its activation has completed.

List Workload Resource Groups of an Ensemble

Use the **List Workload Resource Groups of an Ensemble** operation to list the workload resource groups within the target ensemble.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/workload-resource-groups

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a list of workload resource groups.

Query parameters

Name	Type	Rqd/Opt	Description
`name`	String	Optional	A filter pattern (regular expression) that limits returned objects to those that have a matching **name** property. If matches are found, the response is an array with all workload resource groups that match. If a match is not found, the response is an empty array.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`workloads`	Array of objects	Array of nested workload-info objects as described in the next table.

workloads-info object

Field name	Type	Description
`object-uri`	String/URI	The canonical URI path of the workload object, in the form `/api/workload-resource-groups/{`*`workload-id`*`}`.
`object-id`	String	The object-id of the workload object. This string is the *{workload-id}* portion of the URI path provided by the **object-uri** field.
`name`	String	The display name of the workload object.
`status`	String Enum	The status property of the Workload Resource Group object.

Description

The **List Workload Resource Groups of an Ensemble** operation lists the workload resource groups that belong to the ensemble targeted by the request URI. The object URI, status, object ID, and display name are provided for each.

If the **name** query parameter is specified, the returned list is limited to the workload resource group in the ensemble that has a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

A workload resource group is included in the list only if the API user has object-access permission for that object. An error response is returned if the primary HMC does not manage the target ensemble or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Object-access permission to all workload objects to be included in the result.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The **object-id** in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
GET /api/ensembles/b58f0846-8ef7-11df-bb72-00215ef9b504/workload-resource-groups HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
```

Figure 185. List Workload Resource Groups of an Ensemble: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Mon, 28 Nov 2011 03:52:05 GMT
content-type: application/json;charset=UTF-8
content-length: 452
{
   "workloads": [
      {
         "name": "Default",
         "object-id": "b65a3066-8ef7-11df-95c2-00215ef9b504",
         "object-uri": "/api/workload-resource-groups/b65a3066-8ef7-11df-95c2-00215ef9b504"
      },
      {
         "name": "SS-Web-Store-Workload",
         "object-id": "546a3398-1974-11e1-999c-00215e6a0c26",
         "object-uri": "/api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26"
      }
   ]
}
```

Figure 186. List Workload Resource Groups of an Ensemble: Response

Get Workload Resource Group Properties

Use the **Get Workload Resource Group Properties** operation to retrieve the properties of a single workload object that is designated by its **object-id**. This operation returns the properties of a single workload resource group in the ensemble.

HTTP method and URI

GET /api/workload-resource-groups/{*workload-id*}

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are requesting properties.

Response body contents

On successful completion, the response body is a JSON object provides the current values of the properties for the workload object as defined in "Data model" on page 370. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Note that the **virtual-servers** and **virtual-server-groups** properties are omitted from this response schema. They are available through their own operations: "List Virtual Servers of a Workload Resource Group" on page 383 and "List Groups of Virtual Servers of a Workload Resource Group" on page 389.

Description

The **Get Workload Resource Group Properties** operation returns the current properties for the workload object specified by *{workload-id}*, as defined in "Response body contents."

An error response is returned if the targeted workload resource group does not exist or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirement:

- Object-access permission to the workload object passed in the request URI.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26 HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
```

Figure 187. Get Workload Resource Group Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Mon, 28 Nov 2011 05:09:12 GMT
content-type: application/json;charset=UTF-8
content-length: 1244
{
   "active-perf-policy": {
      "activation-status": "active",
      "element-id": "160c563e-197f-11e1-8914-00215e6a0c26",
      "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
      performance-policies/160c563e-197f-11e1-8914-00215e6a0c26",
      "name": "Prime Shift"
   },
   "category": "Production",
   "class": "workload-resource-group",
   "custom-perf-policies": [
      {
         "activation-status": "active",
         "element-id": "160c563e-197f-11e1-8914-00215e6a0c26",
         "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
       performance-policies/160c563e-197f-11e1-8914-00215e6a0c26",
         "name": "Prime Shift"
      }
   ],
   "default-perf-policy": {
      "activation-status": "not-active",
      "element-id": "13ec9170-197f-11e1-8914-00215e6a0c26",
      "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
      performance-policies/13ec9170-197f-11e1-8914-00215e6a0c26",
      "name": "Default"
   },
   "description": "Spacely Sprockets Web Store Workload",
   "is-default": false,
   "is-locked": false,
   "name": "SS-Web-Store-Workload",
   "object-id": "13de1bfe-197f-11e1-8914-00215e6a0c26",
   "object-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26",
   "parent": "/api/ensembles/b58f0846-8ef7-11df-bb72-00215ef9b504",
   "perf-activation-node-count": 2,
   "perf-activation-status": "active"
```

Figure 188. Get Workload Resource Group Properties: Response

Create Workload Resource Group

Use the **Create Workload Resource Group** operation to create a new custom workload resource group in an ensemble. The new workload resource group must be uniquely named within the ensemble.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/workload-resource-groups

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you want to create a new workload resource group.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`name`	String (1-64)	Required	The name to give the new workload resource group, as described in "Data model" on page 370. The passed name must be unique among all other workload resource groups currently in the ensemble.
`description`	String (0-256)	Optional	The description to give the new workload resource group, as described in "Data model" on page 370.
`category`	String (0-32)	Optional	The category to give the new workload resource group, as described in "Data model" on page 370.

Response body contents

On successful completion, the response body is a JSON object with the following field:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the workload object, in the form **`/api/workload-resource-groups/{`*`workload-id`*`}`**.

Description

The **Create Workload Resource Group** operation creates a new custom workload resource group in the ensemble identified by *{ensemble-id}*. On successful execution, the workload resource group is created and added to the ensemble and status code 201 is returned with a response body containing a reference to the new workload resource group.

The **Create Workload Resource Group** operation returns an empty workload resource group that has a default performance policy but does not contain any virtual servers or user-defined groups. To create a fully functional workload resource group, you also need to use these additional operations, in sequence, to minimize the number of policy activations on the virtual servers:

1. "Create Performance Policy" on page 405– Use this operation to create all custom performance policies desired for the new workload resource group. You can omit this step if you do not want to use any custom performance policies.
2. "Activate Performance Policy" on page 412– Use this operation to set the active policy for the new workload resource group. You can omit this step if you want the default performance policy to be the active policy.
3. "Add Virtual Server to a Workload Resource Group" on page 386 or "Add Group of Virtual Servers to a Workload Resource Group" on page 391– Use one or both of these operations to add all of the virtual servers and groups that you want to be members of the new workload resource group. As the virtual servers or groups are added, the active performance policy of the workload resource group is applied to them.

An error response is returned:

- If the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements" on page 379.
- If the request body is not valid. The request body is validated against the schema described in "Request body contents."

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **New Workload Resource Group** task.

In addition, the target ensemble must be at the Automate entitlement level.

HTTP status and reason codes

On successful completion, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents" on page 378.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See"Common request validation reason codes" on page 18 for a list of the possible reason codes.
	65	The name of the new workload resource group is not unique.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
	3	The ensemble is not operating at the management entitlement level required to perform this operation (Automate).
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/b58f0846-8ef7-11df-bb72-00215ef9b504/workload-resource-groups HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 73
{
   "description": "New Workload Resource Group",
   "name": "SS-New-Workload"
}
```

Figure 189. Create Workload Resource Group: Request

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26
cache-control: no-cache
date: Mon, 28 Nov 2011 03:52:01 GMT
content-type: application/json;charset=UTF-8
content-length: 83
{
   "object-uri": "/api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26"
}
```

Figure 190. Create Workload Resource Group: Response

Delete Workload Resource Group

Use the **Delete Workload Resource Group** operation to remove a workload resource group from an ensemble, and deactivate its active policy against its virtual servers. Any virtual server that is not contained by another custom workload resource group is moved into the default workload resource group.

HTTP method and URI

DELETE /api/workload-resource-groups/{*workload-id*}

In this request, the URI variable *{workload-id}* is the object ID of the workload object that you want to remove.

Description

The **Delete Workload Resource Group** operation deletes the workload object specified by *{workload-id}* from its ensemble.

On successful completion, all virtual servers in the workload resource group are removed from it, the workload resource group is removed from the ensemble, and status code 204 (No Content) is returned without supplying a response body. See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for details about managing virtual servers in workload resource groups.

An error response is returned if the targeted workload object does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also returns an error response because the default workload resource group cannot be deleted.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the ensemble object that owns the workload.
- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Delete Workload Resource Group** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned. Otherwise, the following HTTP status codes are returned for the indicated errors.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	61	The operation cannot be performed because one or more virtual servers in the workload resource group designated by the request URI are currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26 HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
```

Figure 191. Delete Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:06 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 192. Delete Workload Resource Group: Response

Update Workload Resource Group

Use the **Update Workload Resource Group** operation to modify simple writeable properties of a workload object.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}

In this request, the URI variable *{workload-id}* is the object ID of the workload object that you are modifying.

Request body contents

The request body is a JSON object with one or more of the following fields. You are required to supply only those fields that you want to change.

Field name	Type	Rqd/Opt	Description
name	String (1-64)	Optional	The new name to give the workload resource group, as described in "Data model" on page 370. The passed name must be unique among all other workload resource groups currently in the ensemble.
description	String (0-256)	Optional	The new description to give the workload resource group, as described in "Data model" on page 370.
category	String (0-32)	Optional	The new category to give the workload resource group, as described in "Data model" on page 370.
acceptable-status	Array of String Enum	Optional	An array of values that define what causes the workload's status to be compliant as described in "Data model" on page 370.
compliant-perf-status	Array of String Enum	Optional	An array of values that define what **perf-status** values cause the workload's status to be compliant as described in "Data model" on page 370.

Description

The **Update Workload Resource Group** operation updates one or more simple writeable properties of the workload object identified by *{workload-id}*. These properties are those that are not modified through other specific operations, as noted in "Request body contents" on page 381.

The request body is validated against the schema described in "Request body contents" on page 381. On successful execution, the workload object is updated with the supplied property values and status code 204 (No Content) is returned without supplying a response body. Notifications for these property changes are sent asynchronously to this operation.

An error response is returned if the targeted workload resource group does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also invokes an error response because its name, description, and category cannot be modified.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Workload Resource Group Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned. Otherwise, the following HTTP status codes are returned for the indicated errors.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
	65	The new name given to the workload resource group is not unique.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object id in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26 HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 114
{
   "category": "Production",
   "description": "Spacely Sprockets Web Store Workload",
   "name": "SS-Web-Store-Workload"
}
```

Figure 193. Update Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:01 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 194. Update Workload Resource Group: Response

List Virtual Servers of a Workload Resource Group

The **List Virtual Servers of a Workload Resource Group** operation lists the virtual servers in the passed workload. This is the way to get the information corresponding to the **virtual-servers** and **directly-added-virtual-servers** properties of a workload, as it is omitted from the standard GET operation.

HTTP method and URI

GET /api/workload-resource-groups/{*workload-id*}/virtual-servers

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are requesting a list of virtual servers.

Query parameters:

Field name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property. If matches are found, the response will be an array with information for all virtual servers that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`virtual-servers`	Array of objects	Array of nested virtualserver-info objects as described in the next table.

Each nested virtualserver-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the virtual server object.
`name`	String	The current value of the **name** property of the virtual server object.
`status`	String Enum	The current value of the **status** property of the virtual server object.
`type`	String Enum	The value of the **type** property of the virtual server object.
`inclusion-type`	Array of String Enum	The reason(s) this virtual server is included in the workload resource group. The possible values are as follows: • **"direct"**: The virtual server was added directly to the workload resource group. • **"custom-group"**: The virtual server was added to the workload resource group due to its membership in one or more user-defined groups which belong to the workload resource group.

Description

The **List Virtual Servers of a Workload Resource Group** operation lists the virtual servers that belong to the workload resource group targeted by the request URI. The object URI, display name, status, type, and inclusion-type information are provided for each. The inclusion-type information details how the virtual server was added to the workload resource group, as described in "Response body contents."

An error response is returned if the ensemble does not contain the targeted workload or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirement:
- Object-access permission to the workload object passed in the request URI.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/workload-resource-groups/29354e42-20db-11e1-9e8d-00215e6a0c26/virtual-servers HTTP/1.1
x-api-session: 4sz56y4z445abfdwva0qsee8wceszpk34chfjfpcgkfg0i5szv
```

Figure 195. List Virtual Servers of a Workload Resource Group: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Wed, 07 Dec 2011 13:55:59 GMT
content-type: application/json;charset=UTF-8
content-length: 537
{
   "virtual-servers": [
      {
         "inclusion-type": [
            "custom-group"
         ],
         "name": "SS-Web-Svr-3",
         "object-uri": "/api/virtual-servers/1cc18bee-20db-11e1-b49e-00262df332b3",
         "status": "not-operating",
         "type": "x-hyp"
      },
      {
         "inclusion-type": [
            "direct"
         ],
         "name": "SS-Web-Svr-2",
         "object-uri": "/api/virtual-servers/1a4f490a-20db-11e1-b953-00262df332b3",
         "status": "not-operating",
         "type": "x-hyp"
      },
      {
         "inclusion-type": [
            "direct",
            "custom-group"
         ],
         "name": "SS-Web-Svr-1",
         "object-uri": "/api/virtual-servers/180b51de-20db-11e1-b357-00262df332b3",
         "status": "not-operating",
         "type": "x-hyp"
      }
   ]
}
```

Figure 196. List Virtual Servers of a Workload Resource Group: Response

Add Virtual Server to a Workload Resource Group

Use the **Add Virtual Server to a Workload Resource Group** operation to add a virtual server to the target workload resource group.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/operations/add-virtual-server

In this request, the URI variable *{workload-id}* is the object ID of the workload object to which you want to add a virtual server.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`virtual-server`	String/URI	The canonical URI identifying the virtual server object to be added to the target workload resource group.

Description

The **Add Virtual Server to a Workload Resource Group** operation adds a virtual server directly to the workload resource group targeted by the request URI. See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for details about managing virtual servers in workload resource groups.

A virtual server can become a member of a workload resource group through two ways: by adding the virtual server individually or by adding a custom group that contains the virtual server. The way in which the virtual server was added has an effect on the outcome of this operation:

- If the virtual server is not already a member of the workload resource group, the virtual server is added and a property notification is sent asynchronously to this request. If the virtual server was previously in the default workload resource group, the virtual server is removed automatically from the default workload resource group.
- If the virtual server is already directly defined to the workload resource group (that is, the virtual server was added individually at an earlier time), the request is ignored with a successful return code.
- If the virtual server is already a member of the workload resource group through its membership in a group that already belongs to the workload resource group, the **virtual-servers** property does not change. However, it is now directly defined to the workload resource group such that if the virtual server is removed from the group later, it would remain in the workload resource group.

An error response is returned if the targeted workload resource group or virtual server does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also invokes an error response because you cannot directly modify the default workload resource group.

The request body is validated against the schema described in "Request body contents." If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the workload object passed in the request URI.
- Object access permission to the virtual server object passed in the request body.
- Action/task permission to the **Workload Resource Group Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.
	2	The **object-id** in the URI of the virtual server object in the request body does not designate an existing object, or the API user does not have access permission to it.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	61	The operation cannot be performed because the virtual server to be added is currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26/operations/add-virtual-server HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 79
{
   "virtual-server": "/api/virtual-servers/ecd06dfc-1972-11e1-8879-00262df332b3"
}
```

Figure 197. Add Virtual Server to a Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:04 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 198. Add Virtual Server to a Workload Resource Group: Response

Remove Virtual Server from a Workload Resource Group

Use the **Remove Virtual Server from a Workload Resource Group** operation to remove a virtual server from the target workload resource group.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/operations/remove-virtual-server

In this request, the URI variable *{workload-id}* is the object ID of the workload object from which you want to remove the virtual server.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`virtual-server`	String/URI	The canonical URI identifying the virtual server object to be removed from the targeted workload resource group.

Description

The **Remove Virtual Server from a Workload Resource Group** operation removes a virtual server from being directly defined to the workload resource group targeted by the request URI. See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for details about managing virtual servers in workload resource groups.

A virtual server can become a member of a workload resource group through two ways: by adding the virtual server individually or by adding a custom group that contains the virtual server. In some cases, a virtual server can be defined twice to the same workload resource group. The way in which the virtual server was added to the workload resource group has an effect on the outcome of this operation:

- If the virtual server was added individually to the workload resource group and is not also a member of a custom group that is already defined to this workload resource group, the virtual server is removed. A property change notification is sent asynchronously to this request.
- If the virtual server was added individually and as a member of a custom group in the workload resource group, the virtual server remains in the workload resource group only as a member of the custom group. In this case, a property change notification is not sent.
- If the virtual server is not already a member of the workload resource group (either individually or through a custom group), the request is ignored with a successful return code.

An error response is returned if the targeted workload resource group or virtual server does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also invokes an error response because you cannot directly modify the default workload resource group.

The request body is validated against the schema described in "Request body contents." If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the workload object passed in the request URI.
- Object access permission to the virtual server object passed in the request body.
- Action/task permission to the **Workload Resource Group Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.
	2	The **object-id** in the URI of the virtual server object in the request body does not designate an existing object, or the API user does not have access permission to it.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	61	The operation cannot be performed because the virtual server to be removed is currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26/operations/
  remove-virtual-server HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 79
{
   "virtual-server": "/api/virtual-servers/f1f581f4-196e-11e1-a344-00262df332b3"
}
```

Figure 199. Remove Virtual Server from a Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:06 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 200. Remove Virtual Server from a Workload Resource Group: Response

List Groups of Virtual Servers of a Workload Resource Group

The **List Groups of Virtual Servers of a Workload Resource Group** operation lists the user-defined managed object groups in the passed workload. This is a way to just get the information maintained in the **virtual-server-groups** property of a workload.

HTTP method and URI

GET /api/workload-resource-groups/{*workload-id*}/virtual-server-groups

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are requesting a list of groups.

Query parameters:

Field name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern to limit returned objects to those that have a matching **name** property. If matches are found, the response will be an array with information for all groups that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body is a JSON object containing the following fields:

Field name	Type	Description
`virtual-server-groups`	Array of String/URI	An array of user-defined managed object groups in the workload resource group, each identified by its URI.

Description

The **List Groups of Virtual Servers of a Workload Resource Group** operation lists the user-defined managed object groups that belong to the workload resource group targeted by the request URI. The group object URI is provided for each.

A group is included in the list only if you meet the requirements listed in "Authorization requirements"; otherwise, an error is returned. An error also is returned if the ensemble does not contain the targeted workload resource group.

Authorization requirements

This operation has the following authorization requirement:

- Object-access permission to the workload object passed in the request URI.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object-access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26/virtual-server-groups HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
```

Figure 201. List Groups of Virtual Servers of a Workload Resource Group: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Mon, 28 Nov 2011 03:52:05 GMT
content-type: application/json;charset=UTF-8
content-length: 78
{
   "virtual-server-groups": [
      "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341"
   ]
}
```

Figure 202. List Groups of Virtual Servers of a Workload Resource Group: Response

Add Group of Virtual Servers to a Workload Resource Group

Use the **Add Group of Virtual Servers to a Workload Resource Group** operation to add a user-defined managed object group to the target workload resource group. Any virtual servers that the group contains are added as virtual servers of the workload resource group.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/operations/add-virtual-server-group

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are requesting a list of groups.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`virtual-server-group`	String/URI	The canonical URI identifying the user-defined group object to be added to the targeted workload resource group.

Description

The **Add Group of Virtual Servers to a Workload Resource Group** operation adds a user-defined group to the workload resource group targeted by the request URI. Through this operation, the virtual servers that belong to the group are added automatically to the workload. See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for details about managing virtual servers in workloads.

If the group is already defined to the workload, the request is ignored with a successful return code. If the group is not already defined to the workload resource group, the group is added and a property change notification for the **virtual-server-groups** property is sent asynchronously. All virtual servers in

the group are added to the workload resource group, and another property change notification is sent asynchronously for the **virtual-servers** property including the virtual servers that are entirely new to the workload. If any of these virtual servers previously belonged to the default workload resource group, they are removed automatically from the default workload resource group.

An error response is returned if the targeted workload resource group or custom group does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also results in an error because the default workload is not directly mutable. Pattern match groups are not supported so specifying them in an **Add Group of Virtual Servers to a Workload** operation results in an error.

The request body is validated against the schema described in "Request body contents" on page 391. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the workload object passed in the request URI.
- Object-access permission to the group object passed in the request body.
- Object-access permission to all virtual servers contained by the group.
- Action/task permission to the **Workload Resource Group Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
	61	The **object-id** of the group to be added in the request body is that of a pattern match group.
	62	The group to be added contains virtual servers to which the API user does not have access.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.
	2	The **object-id** in the URI of the group object in the request body does not designate an existing object, or the API user does not have access permission to it.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	62	The operation cannot be performed because the group to be added contains one or more virtual servers that are currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26/operations/
  add-virtual-server-group HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 76
{
   "virtual-server-group": "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341"
}
```

Figure 203. Add Group of Virtual Servers to a Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:05 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 204. Add Group of Virtual Servers to a Workload Resource Group: Response

Remove Group of Virtual Servers from a Workload Resource Group

Use the **Remove Group of Virtual Servers from a Workload Resource Group** operation to remove a user-defined managed object group from the target workload resource group. All virtual servers in the group are removed from the workload resource group, unless the virtual server belongs to another group that is associated with the same workload resource group, or is directly added.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/operations/remove-virtual-server-group

In this request, the URI variable *{workload-id}* is the object ID of the workload object from which you are removing the group.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Description
`virtual-server-group`	String/URI	The canonical URI identifying the user-defined group object to be removed from the targeted workload resource group.

Description

The **Remove Group of Virtual Servers from a Workload Resource Group** operation removes a user-defined group from the workload resource group targeted by the request URI. Through this operation, any virtual servers in the group are removed as well, unless they also belong to another group that is associated with this workload resource group. See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for details about managing virtual servers in workload resource groups.

If the group is not defined to the workload resource group, then the request is ignored with a successful return code. If the group is in the workload resource group, then it is removed and a property change notification for the **virtual-server-groups** property is sent asynchronously. Virtual servers in the group that are not also members of another group in the workload resource group, or directly added, are removed from the workload resource group, and another property change notification is sent asynchronously for the **virtual-servers** property. If any of those virtual servers do not belong to another custom workload resource group, they are placed in the default workload resource group.

An error response is returned if the targeted workload resource group or group does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also invokes an error response because you cannot directly modify the default workload resource group.

The request body is validated against the schema described in "Request body contents" on page 393. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the workload object passed in the request URI.
- Object access permission to the group object passed in the request body.
- Object access permission to all virtual servers contained by the group.
- Action/task permission to the **Workload Resource Group Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
	62	The group to be removed contains virtual servers to which the API user does not have access.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.
	2	The **object-id** in the URI of the virtual server object in the request body does not designate an existing object, or the API user does not have access permission to it.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	62	The operation cannot be performed because the group to be removed contains one or more virtual servers that are currently busy performing some other operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/546a3398-1974-11e1-999c-00215e6a0c26/operations/
  remove-virtual-server-group HTTP/1.1
x-api-session: 466ske8jt8waeavxcs5rc26gsfjqjfs6aji8qbj7jgne6yrdbq
content-type: application/json
content-length: 76
{
   "virtual-server-group": "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341"
}
```

Figure 205. Remove Group of Virtual Servers from a Workload Resource Group: Request

```
204 No Content
date: Mon, 28 Nov 2011 03:52:06 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 206. Remove Group of Virtual Servers from a Workload Resource Group: Response

Performance policy object

Performance policy objects are elements that define performance goals for virtual servers in a workload resource group. A performance policy consists mainly of an importance level and an ordered list of service classes that define the goals. See "Service class nested object" on page 398 for more details on service classes.

Any number of custom performance policy objects can be defined for a workload resource group. However, exactly one is active at a time. Every workload resource group contains an immutable default performance policy that is active if a custom workload policy is not activated. Note that changing any property of an active performance policy automatically causes its reactivation. Because activating a performance policy can fail, it follows then that updating properties of an active performance policy can fail. See "Activate Performance Policy" on page 412 for further details.

Data model

This element includes the following type-specific properties. For definitions of the qualifier abbreviations in the following table, see "Property characteristics" on page 32.

Table 75. Performance policy object: type-specific properties

Name	Qualifier	Type	Description of specialization
element-uri	—	String/URI	The canonical URI path for a performance policy is qualified by its workload object and is of the form `/api/workload-resource-groups/{workload-id}/performance-policies/{policy-id}` where *{workload-id}* is the value of the **object-id** property of the parent workload object and *{policy-id}* is the value of the **element-id** property of the policy

Table 75. Performance policy object: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
element-id	—	String (36)	The unique identifier for the performance policy instance. The unique identifier is in the form of a UUID.
parent	—	String/URI	The parent of a performance policy is its owning workload resource group, and so the **parent** value is the canonical URI path for the workload resource group.
class	—	String	The class of a performance policy object is "performance-policy".
name	(w) (pc)	String (1-64)	The display name specified for the performance policy, which can be up to 64 characters made up of alphanumeric characters, blanks, periods, underscores, or dashes. Names must start with an alphabetic character and end with an alphabetic character, numeric character, underscore, period, or dash. Names must be unique to other existing performance policies in the workload resource group.
description	(w) (pc)	String (0-256)	Arbitrary text describing the performance policy in up to 256 characters.
is-default	—	Boolean	This value is used to identify the default performance policy object. It is true for the default policy and false for all other (custom) policies in the workload resource group.
importance	(w) (pc)	String Enum	The importance value assigned to the performance policy, which is one of the following: • **"highest"** • **"high"** • **"medium"** • **"low"** • **"lowest"** See the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607, for detailed information on the **importance** property values.
custom-service-classes	(w) (pc)	Array of objects (0-99)	The ordered list of custom service classes in the performance policy. Each service class will be an object in the form of a service class object, as described in "Service class nested object" on page 398. This list includes only custom service classes and not the default service class for the policy. This array can contain from 0 to 99 entries.
default-service-class	—	Object	The object describing the default service class for the performance policy, in the form of the service class object, as described in "Service class nested object" on page 398.
revision	(pc)	Integer (1-n)	The revision count of the performance policy, which is the number of times the policy has been modified. The initial value for a new performance policy is 1.

Table 75. Performance policy object: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
activation-status	(pc)	String Enum	The activation status of the performance policy, which is one of the following values: • **"not-active"** – the performance policy is not currently the active policy for the workload resource group • **"in-progress"** – the performance policy is currently being activated for the workload resource group • **"active"** – the performance policy is currently the active policy for the workload resource group, and its activation has completed
last-activation-requested-date	(pc)	Integer	The standard date/time value indicating the time of the last activation request of the policy. If the policy has never been active or is the default policy of the default workload resource group, this value is negative. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
last-activation-completed-date	(pc)	Integer	The standard date/time value indicating the time the policy last completed activation. If the policy has never been active or is the default policy of the default workload resource group, this value is negative. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
last-activated-by	(pc)	String	The user name used for the last activation request. This value is the empty string if the policy has never been activated or is the default policy of the default workload resource group.
last-modified-date	(pc)	Integer	The standard date/time value indicating the time the policy was last modified. If the policy has never been modified, this property is equal to the **created-date** property. The value is negative for all default policies. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
last-modified-by	(pc)	String	The user name used to last modify the policy. If the policy has never been modified, this property is equal to the **created-by** property. The value is the empty string for all default policies.
created-date	—	Integer	The standard date/time value indicating the time the policy was created. The value is negative for all default policies. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.

Table 75. Performance policy object: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
created-by	—	String	The user name used to create the policy. The value is the empty string for all default policies.

Service class nested object

A service class object is a nested object of a performance policy object. A performance policy can contain zero or more ordered service classes, and contains a default service class. Service classes use classification rules to classify and apply performance goals to virtual servers in the workload resource group. Service classes are positional; during classification, zManager searches service classes from low-ordered to high-ordered service class to find a match for a virtual server. The default service class applies for any virtual server that does not match a custom service class.

You cannot directly change a service class object. When any property of a service class must be changed, zManager creates a new service class object to replace the existing one in the policy object. For this reason, service classes are not supported as objects in the standard sense, but rather as simple nested objects of a performance policy. Because of this fact, direct operations and notifications are not supported for service class objects.

The following properties are supported:

Table 76. Performance policy object: Service class nested object properties

Field name	Type	Rqd/Opt	Description
name	String (1-64)	Required	The display name specified for the service class, which can be up to 64 characters made up of alphanumeric characters, blanks, periods, underscores, or dashes. Names must start with an alphabetic character and end with an alphabetic character, numeric character, underscore, period, or dash. Names must be unique to other existing service classes in the performance policy.
description	String (0-256)	Optional	Arbitrary text describing the performance policy in up to 256 characters.
type	String Enum	Required	This identifies the type of the service class object: • **"server"** – the service class type targeting specific virtual servers
goal-type	String Enum	Required	This identifies the type of performance goal for the service class, which must be one of the following: • **"velocity"** – the service class goal is based on a velocity value as given in the velocity property • **"discretionary"** – the service class goal is up to the discretion of zManager
business-importance	String Enum	Optional*	This field identifies the business importance level assigned to the service class, which must be one of the following: • **"highest"** • **"high"** • **"medium"** • **"low"** • **"lowest"** * - A business importance is required for the velocity goal type, but is not used for a discretionary goal type.

Table 76. Performance policy object: Service class nested object properties (continued)

Field name	Type	Rqd/Opt	Description
velocity	String Enum	Optional*	This field identifies the velocity goal value of the service class, which must be one of the following: • **"fastest"** • **"fast"** • **"moderate"** • **"slow"** • **"slowest"** * - A velocity value is only required for the velocity goal type.
classification-rule	Object	Required	The rule used to filter the virtual servers for which the service class goal applies. The value must be a JSON object, as described in "Classification rule nested object."

Classification rule nested object

A classification rule object is a nested object within a service class object. It is a recursive object used to define logical filter statements to be applied to the virtual servers of the workload resource group. If a virtual server passes the classification rule, the service class goal is applied.

Table 77. Performance policy object: classification rule nested object properties

Field name	Type	Rqd/Opt	Description
type	String Enum	Required	This field identifies the type of classification rule object, which must be one of the following: • **"and"** – for this rule to be true, both of the two rules referenced by this rule must be true • **"or"** – for this rule to be true, only one of the two rules referenced by this rule must be true • **"rule"** – the rule defines a filter that resolves to true or false based on its filter pattern against a virtual server If you specify **"and"** or **"or"**, exactly two classification rule objects must be nested inside this classification rule object so they can be logically compared. If you specify **"rule"**, exactly one filter object must be nested inside this object.
classification-rule-1	Object	Required*	The first of two classification rule objects that must be nested inside this classification rule object if they are to be logically compared as defined by the **type** property. If the **type** is **"rule"**, this property is not supported. * - The **type** property value determines whether this field is required.
classification-rule-2	Object	Required*	The second of two classification rule objects that must be nested inside this classification rule object if they are to be logically compared as defined by the **type** property. If the **type** is **"rule"**, this property is not supported. * - The **type** property value determines whether this field is required.
filter	Object	Required*	A filter object defining the filter statement if the classification rule type is **"rule"**. If the **type** is **"and"** or **"or"**, this property is not supported. * - The **type** property value determines whether this field is required.

Filter nested object

A filter object is a nested object within a classification rule object whose type is **"rule"**. The filter object defines a logical statement used to filter on properties of a virtual server.

Table 78. Performance policy object: filter nested object properties

Field name	Type	Rqd/Opt	Description
type	String Enum	Required	This field identifies the type of the filter object, which must be one of the following: • **"hostname"** – the filter value is matched against the hostname of the virtual server • **"virtual-server-name"** – the filter value is matched against the name of the virtual server • **"os-type"** – the filter value is matched against the operating system type of the virtual server • **"os-level"** – the filter value is matched against the operating system level of the virtual server • **"os-name"** – the filter value is matched against the operating system name of the virtual server
operation	String Enum	Required	This field identifies the logical filter operation, which must be one of the following: • **"string-match"** – the filter value must match the property defined by the filter type • **"string-not-match"** – the filter value must not match the property defined by the filter type
value	String (1-255)	Required	A string to match the property against. Only patterns "." (to match any character) and ".*" (to match any character zero or more times) are currently supported and ".*" may only be used at the end of the filter value string. The following characters must be escaped to be used directly: +-()[]$^.*\".

Notifications of property changes to performance policies

Notifications of property changes to performance policy objects are similar to notifications of changes to workload objects.

Because performance policy objects are sub-objects of workload objects, creating or deleting a performance policy does not result in an inventory notification. Instead, these operations result in the appropriate property notification for the target workload object. However, updating performance policy properties results in property notifications for the policy itself. In these cases, the notification header contains an **element uri** and an **element id** to indicate the performance policy that changed.

List Performance Policies

Use the **List Performance Policies** operation to list the performance policies within the target workload resource group.

HTTP method and URI

GET /api/workload-resource-groups/{*workload-id*}/performance-policies

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are requesting a list of performance policies.

Query parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	Filter pattern (regular expression) to limit returned objects to those that have a matching **name** property. If a match is found, the response is an array with all policies that match. If no match is found, the response is an empty array.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`perf-policies`	Array of objects	Array of nested perf-policy-info objects as the following table.

Each nested perf-policy-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the performance policy object, in the form `/api/workload-resource-groups/{`*`workload-id`*`}/performance-policies/{`*`policy-id`*`}`.
`element-id`	String	The **element-id** of the performance policy object. This field value is the *{policy-id}* portion of the URI path provided by the **element-uri** field.
`name`	String	Display name of the performance policy object.
`activation-status`	String Enum	The status of the performance policy object, which must be one of the following values: • **"not-active"** – the performance policy is not currently the active policy for the workload resource group • **"in-progress"** – the performance policy is currently being activated for the workload resource group • **"active"** – the performance policy is currently the active policy for the workload resource group, and its activation has completed

Description

The **List Performance Policies** operation lists the performance policies that belong to the workload resource group targeted by the request URI. The element URI, element ID, display name, and status are provided for each.

If the **name** query parameter is specified, the returned list is limited to the policies in the workload resource group that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

An error response is returned if the HMC does not manage the targeted workload resource group or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirement:

- Object-access permission to the workload object passed in the request URI. A policy is included in the list only if you also have object-access permission for that policy object.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The **object-id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to it.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
  performance-policies HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
```

Figure 207. List Performance Policies: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Mon, 28 Nov 2011 05:09:12 GMT
content-type: application/json;charset=UTF-8
content-length: 509
{
   "perf-policies": [
      {
         "activation-status": "not-active",
         "element-id": "13ec9170-197f-11e1-8914-00215e6a0c26",
         "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
       performance-policies/13ec9170-197f-11e1-8914-00215e6a0c26",
         "name": "Default"
      },
      {
         "activation-status": "active",
         "element-id": "160c563e-197f-11e1-8914-00215e6a0c26",
         "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
       performance-policies/160c563e-197f-11e1-8914-00215e6a0c26",
         "name": "Prime Shift"
      }
   ]
}
```

Figure 208. List Performance Policies: Response

Get Performance Policy Properties

Use the **Get Performance Policy Properties** operation to retrieve the properties of a single performance policy.

HTTP method and URI

GET /api/workload-resource-groups/{*workload-id*}/performance-policies/{*policy-id*}

URI variables

Variable	Description
{workload-id}	Object ID of the workload object for the workload resource group to which the performance policy is defined.
{policy-id}	Element ID of the performance policy object for which properties are to be obtained.

Response body contents

On successful completion, the response body is a JSON object that provides the current values of the properties for the performance policy object as defined in "Data model" on page 395. Field names and data types in the JSON object are the same as the property names and data types defined in the data model.

Description

The **Get Performance Policy Properties** operation returns the current properties for the performance policy object specified by *{policy-id}*.

An error response is returned if the targeted workload resource group or policy does not exist or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirement:
- Object-access permission to the workload object passed in the request URI.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
	62	The **element id** in the URI (*{policy-id}*) does not designate an existing performance policy object in the workload resource group.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
  performance-policies/160c563e-197f-11e1-8914-00215e6a0c26 HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
```

Figure 209. Get Performance Policy Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Mon, 28 Nov 2011 05:09:12 GMT
content-type: application/json;charset=UTF-8
content-length: 1586
{
   "activation-status": "active",
   "class": "performance-policy",
   "created-by": "ENSADMIN",
   "created-date": 1322456941929,
   "custom-service-classes": [
      {
         "business-importance": "highest",
         "classification-rule": {
            "filter": {
               "operation": "string-match",
               "type": "virtual-server-name",
               "value": "SS\\-Premium\\-Web\\-Svr\\-.*"
            },
            "type": "rule"
         },
         "description": "",
         "goal-type": "velocity",
         "name": "Premium Class",
         "type": "server",
         "velocity": "fastest"
      },
      {
         "business-importance": "high",
         "classification-rule": {
            "filter": {
               "operation": "string-match",
               "type": "virtual-server-name",
               "value": "SS\\-Web\\-Svr\\-.*"
            },
            "type": "rule"
         },
         "description": "",
         "goal-type": "velocity",
         "name": "Regular Class",
         "type": "server",
         "velocity": "moderate"
      }
   ],
```

Figure 210. Get Performance Policy Properties: Response (Part 1)

```
"default-service-class": {
      "business-importance": "medium",
      "classification-rule": {
         "filter": {
            "operation": "string-match",
            "type": "(\\*)",
            "value": "(\\*)"
         },
         "type": "rule"
      },
      "description": "The default workload performance policy service class.",
      "goal-type": "velocity",
      "name": "Default",
      "type": "server",
      "velocity": "moderate"
   },
   "description": "Performance policy for prime shift",
   "element-id": "160c563e-197f-11e1-8914-00215e6a0c26",
   "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
    performance-policies/160c563e-197f-11e1-8914-00215e6a0c26",
   "importance": "highest",
   "is-default": false,
   "last-activated-by": "ENSADMIN",
   "last-activation-completed-date": 1322456944022,
   "last-activation-requested-date": 1322456942144,
   "last-modified-by": "ENSADMIN",
   "last-modified-date": 1322456942090,
   "name": "Prime Shift",
   "parent": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26",
   "revision": 2
}
```

Figure 211. Get Performance Policy Properties: Response (Part 2)

Create Performance Policy

Use the **Create Performance Policy** operation to create a new custom performance policy for a workload resource group. The new policy must be uniquely named within the workload resource group. The policy is inactive until the **Activate Performance Policy** operation is applied to it.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/performance-policies

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are creating a new performance policy.

Request body contents

The request body contains properties used to define the new performance policy, which are the writeable properties of a performance policy. Some properties are optional.

Field name	Type	Rqd/Opt	Description
name	String (1-64)	Required	The name to give the new performance policy, as described in "Data model" on page 395. The passed name must be unique among all other policies currently in the workload resource group.
description	String (0-256)	Optional	The description to give the new performance policy, as described in "Data model" on page 395.

Field name	Type	Rqd/Opt	Description
`importance`	String Enum	Required	The importance value to give the new performance policy, as described in "Data model" on page 395.
`custom-service-classes`	Array of objects (0-99)	Optional	The ordered list of custom service classes in the new performance policy. Each service class is an object in the form of a service class object, as described in "Service class nested object" on page 398. This array can contain from 0 to 99 entries.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the performance policy object, in the form `/api/workload-resource-groups/{`*`workload-id`*`}/performance-policies/{`*`policy-id`*`}`

Description

The **Create Performance Policy** operation creates a new custom performance policy in a workload resource group, identified by *{workload-id}*.

On successful execution, the performance policy is created and added to the workload resource group and status code 201 is returned with a response body containing a reference to the new performance policy object. Note that the new policy is not active until the **Activate Performance Policy** operation is applied to it.

An error response is returned if the targeted workload resource group does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload object also results in an error because the default workload resource group cannot contain any performance policy other than the default performance policy.

The request body is validated against the data model for this object type to ensure that it contains only writeable properties and that the data types of those properties are specified as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **New Performance Policy** task.

HTTP status and reason codes

On successful completion, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
	65	The name of the performance policy is not unique within its workload resource group, or one or more of the defined service class names is reserved or not unique.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
415 (Unsupported Media Type)	0	The request does not include a **Content-Type** header that specifies the request body is of media type **application/xml**.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/performance-policies HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
content-type: application/json
content-length: 101
{
   "description": "Performance policy for prime shift",
   "importance": "highest",
   "name": "Prime Shift"
}
```

Figure 212. Create Performance Policy: Request

```
201 Created
server: zSeries management console API web server / 1.0
location: /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
  performance-policies/160c563e-197f-11e1-8914-00215e6a0c26
cache-control: no-cache
date: Mon, 28 Nov 2011 05:09:01 GMT
content-type: application/json;charset=UTF-8
content-length: 142
{
   "element-uri": "/api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
    performance-policies/160c563e-197f-11e1-8914-00215e6a0c26"
}
```

Figure 213. Create Performance Policy: Response

Delete Performance Policy

Use the **Delete Performance Policy** operation to remove a performance policy from a workload resource group. This operation cannot be performed on an active or default performance policy.

HTTP method and URI

DELETE /api/workload-resource-groups/{*workload-id*}/performance-policies/{*policy-id*}

URI variables

Variable	Description
{workload-id}	Object ID of the workload object whose performance policy object is to be deleted.
{policy-id}	Element ID of the performance policy object that is to be deleted.

Description

The **Delete Performance Policy** operation deletes the performance policy object specified by *{policy-id}* from its workload resource group specified by *{workload-id}*.

On successful execution, the performance policy object is removed from the workload resource group and status code 204 (No Content) is returned without a response body.

An error response is returned if the targeted policy object does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting a default performance policy also results in an error because the default performance policy cannot be deleted. You also cannot delete an active performance policy without first activating another performance policy through the **Activate Performance Policy** operation.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Delete Performance Policy** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned without a response body.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	63	The targeted performance policy is a default performance policy object.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
	62	The **element id** in the URI (*{policy-id}*) does not designate an existing performance policy object in the workload resource group.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	60	The performance policy to be deleted is currently active or in the progress of being activated. The operation can be retried after a different policy has been activated, which causes the **activation-status** of this policy to be **"not-active"**.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/
  performance-policies/160c563e-197f-11e1-8914-00215e6a0c26 HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
```

Figure 214. Delete Performance Policy: Request

```
204 No Content
date: Mon, 28 Nov 2011 05:09:22 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 215. Delete Performance Policy: Response

Update Performance Policy

Use the **Update Performance Policy** operation to modify one or more writeable properties of a performance policy object. Note that updating an active performance policy causes its reactivation.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/performance-policies/{*policy-id*}

URI variables

Variable	Description
{workload-id}	Object ID of the workload object to which the target performance policy is defined.
{policy-id}	Element ID of the performance policy object that is to be modified.

Request body contents

The request body is a JSON object containing one or more of the writeable fields for a performance policy, as described in "Data model" on page 395. You need to supply only those fields that you want to modify.

Description

The **Update Performance Policy** operation updates one or more writeable properties of the performance policy object identified by *{policy-id}*.

On successful execution, the performance policy object is updated with the supplied property values and status code 204 (No Content) is returned without a response body. Notifications for these property changes are sent asynchronously to this operation.

If the performance policy is active at the time of this request, a reactivation of the policy is submitted asynchronously to this operation. Because activation of a performance policy is rejected if another activation request is in progress for the target workload resource group, an update to an active performance policy also can be rejected. In this case, a 409 (Conflict) status code is returned and you can retry the update operation after the first activation completes.

An error response is returned if the targeted workload resource group or performance policy does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default performance policy also results in an error because the default performance policy cannot be directly modified.

The request body is validated against the schema described in "Request body contents" on page 409. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Performance Policy Details** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned without a response body.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	63	The targeted performance policy is a default performance policy object.
	65	The new name given to the performance policy is not unique within its workload resource group, or one or more of the defined service class names are reserved or not unique.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
	62	The **element id** in the URI (*{policy-id}*) does not designate an existing performance policy object in the workload resource group.

HTTP error status code	Reason code	Description
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	60	The performance policy to be updated is currently in the progress of being activated. The operation can be retried after activation has completed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/performance-policies/
  160c563e-197f-11e1-8914-00215e6a0c26 HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
content-type: application/json
content-length: 580
{
   "custom-service-classes": [
      {
         "business-importance": "highest",
         "classification-rule": {
            "filter": {
               "operation": "string-match",
               "type": "virtual-server-name",
               "value": "SS\\-Premium\\-Web\\-Svr\\-.*"
            },
            "type": "rule"
         },
         "goal-type": "velocity",
         "name": "Premium Class",
         "type": "server",
         "velocity": "fastest"
      },
      {
         "business-importance": "high",
         "classification-rule": {
            "filter": {
               "operation": "string-match",
               "type": "virtual-server-name",
               "value": "SS\\-Web\\-Svr\\-.*"
            },
            "type": "rule"
         },
         "goal-type": "velocity",
         "name": "Regular Class",
         "type": "server",
         "velocity": "moderate"
      }
   ]
}
```

Figure 216. Update Performance Policy: Request

```
204 No Content
date: Mon, 28 Nov 2011 05:09:01 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 217. Update Performance Policy: Response

Activate Performance Policy

Use the **Activate Performance Policy** operation to activate a performance policy for a workload resource group. You can activate any performance policy defined to the workload resource group, including the default and currently active policy.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/performance-policies/{*policy-id*}/operations/activate

URI variables

Variable	Description
{workload-id}	Object ID of the workload object to which the target performance policy is defined.
{policy-id}	Element ID of the performance policy object that is to be activated.

Description

The **Activate Performance Policy** operation activates or reactivates a performance policy for a specific workload resource group.

On successful execution, the target policy is active and status code 204 (No Content) is returned without a response body. Notifications for ensuing property changes are sent asynchronously to this operation.

An activation request is not accepted if another activation request is in progress for this workload resource group. In this case, status code 409 (Conflict) is returned and you can retry this operation after the first activation completes.

An error response is returned if the targeted workload resource group or performance policy does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload resource group also results in an error because its default performance policy is permanently active.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Activate Performance Policy** task.

HTTP status and reason codes

On successful completion, HTTP status code 204 (No Content) is returned without a response body.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
	62	The **element id** in the URI (*{policy-id}*) does not designate an existing performance policy object in the workload resource group.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
	60	Performance policy activation is currently in progress on the workload resource group. The operation can be retried after activation has completed.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/13de1bfe-197f-11e1-8914-00215e6a0c26/performance-policies/
  160c563e-197f-11e1-8914-00215e6a0c26/operations/activate HTTP/1.1
x-api-session: 67prscbokwxz6o1bn1q3feysece2q4275agf27uupjnvr98lse
```

Figure 218. Activate Performance Policy: Request

```
204 No Content
date: Mon, 28 Nov 2011 05:09:01 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 219. Activate Performance Policy: Response

Import Performance Policy

Use the **Import Performance Policy** operation to create a new custom performance policy for a workload resource group. This operation is equivalent to the **Create Performance Policy** operation, except the request body is an XML document defining the configuration of the performance policy to be created. The new policy must be uniquely named within the workload resource group. The policy is inactive until the **Activate Performance Policy** operation is applied to it.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/operations/import-performance-policy

In this request, the URI variable *{workload-id}* is the object ID of the workload object for which you are importing a performance policy.

Request body contents

The request body must contain an XML document that describes the performance policy to be created. The XML must conform to the schema described in Appendix A, "XML document structure of a performance policy," on page 677. Note that the same rules apply as for the Create Performance Policy operation, that the name must be unique within the workload.

Because the request body is expected to be in XML format, the request should specify MIME type **application/xml** as the value of the HTTP **Content-Type** header for the request.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the performance policy object, in the form `/api/workload-resource-groups/{`*`workload-id`*`}/performance-policies/{`*`policy-id`*`}`

Description

The **Import Performance Policy** operation imports a performance policy from an XML document and creates a performance policy object, adding it to the workload resource group identified by *{workload-id}*.

On successful execution, the performance policy object is created with the supplied property values, added to the collection of custom performance policies for the workload resource group. Status code 201 (Created) is returned with a response body describing the location of the new performance policy object. Note that the new policy is not active until the **Activate Performance Policy** operation is applied to it.

An error response is returned if the targeted workload does not exist or if you do not have the requirements listed in "Authorization requirements." Targeting the default workload object also results in an error because the default workload resource group cannot contain any performance policy other than the default performance policy.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Import Performance Policy** task.

HTTP status and reason codes

On successful completion, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents."

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	60	The targeted workload resource group is the default workload object.
	65	The name of the performance policy is not unique within its workload resource group, or one or more of the defined service class names is reserved or not unique.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
409 (Conflict)	2	The operation cannot be performed because the object designated by the request URI is currently busy performing some other operation.
415 (Unsupported Media Type)	0	The request does not include a **Content-Type** header that specifies the request body is of media type **application/xml**.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Export Performance Policy

Use the **Export Performance Policy** operation to return the configuration of an existing performance policy in the form of an XML document.

HTTP method and URI

POST /api/workload-resource-groups/{*workload-id*}/performance-policies/{*policy-id*}/operations/export

URI variables

Variable	Description
{workload-id}	Object ID of the workload object for the workload resource group to which the performance policy is defined.
{policy-id}	Element ID of the performance policy object to be exported.

Response body contents

On successful completion, the performance policy is returned in the response body as an XML document that conforms to the schema described in Appendix A, "XML document structure of a performance policy," on page 677.

Description

The **Export Performance Policy** operation returns the configurable properties of an existing performance policy identified by *{policy-id}* in a workload resource group identified by *{workload-id}*. The policy configuration is returned in the form of an XML document. You can save this XML as a backup copy of the policy configuration or modify and import it through the **Import Performance Policy** operation to create a new custom policy.

On successful execution, the performance policy object returned as a XML document with status code 200 (OK).

An error response is returned if the targeted workload resource group or policy does not exist or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the workload object passed in the request URI.
- Action/task permission to the **Export Performance Policy** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 415.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The **object id** in the URI (*{workload-id}*) does not designate an existing workload object, or the API user does not have object access permission to the object.
	62	The **element id** in the URI (*{policy-id}*) does not designate an existing performance policy object in the workload resource group.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/workload-resource-groups/e39812b2-209d-11e1-8bbc-0010184c8334/performance-policies/
   e44af260-209d-11e1-8bbc-0010184c8334/operations/export HTTP/1.1
x-api-session: 2ggnk3nbcxsqi8w8qbjpnl4pqhwgkvdrdu5hw8pjw1o171qu2r
```

Figure 220. Export Performance Policy: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache, no-cache
date: Wed, 07 Dec 2011 06:37:21 GMT
content-type: application/xml;charset=utf-8
content-length: 1515
<?xml version="1.0" encoding="UTF-8"?>

<PerformancePolicy xmlns="http://www.ibm.com/PPM/WorkloadPerformancePolicy-1.0">
  <Name>Prime Shift</Name>
  <Description>Performance policy for prime shift</Description>
  <Version>3.00.00</Version>
  <UI>PPM Editor</UI>
  <WorkloadImportance>Highest</WorkloadImportance>

  <ServiceClasses>

    <ServiceClass>
      <Name>Premium Class</Name>
      <Type>Server</Type>
      <RuleBuilderElement>
        <RuleBuilderElementType>Rule</RuleBuilderElementType>
        <Filter>
          <FilterType>Virtual Server Name</FilterType>
          <FilterOperation>stringMatch</FilterOperation>
          <FilterValue>SS\\-Premium\\-Web\\-Svr\\-\.\*</FilterValue>
        </Filter>
      </RuleBuilderElement>
      <Goal>
        <Velocity>
          <Importance>Highest</Importance>
          <Level>Fastest</Level>
        </Velocity>
      </Goal>
    </ServiceClass>

    <ServiceClass>
      <Name>Regular Class</Name>
      <Type>Server</Type>
      <RuleBuilderElement>
        <RuleBuilderElementType>Rule</RuleBuilderElementType>
        <Filter>
          <FilterType>Virtual Server Name</FilterType>
          <FilterOperation>stringMatch</FilterOperation>
          <FilterValue>SS\\-Web\\-Svr\\-\.\*</FilterValue>
        </Filter>
      </RuleBuilderElement>
      <Goal>
        <Velocity>
          <Importance>High</Importance>
          <Level>Moderate</Level>
        </Velocity>
      </Goal>
    </ServiceClass>

  </ServiceClasses>

</PerformancePolicy>
```

Figure 221. Export Performance Policy: Response

Performance management reports

Through specific APIs described in this topic, you can request zManager to generate historical reports that contain performance data related to specific performance management objects. You can request data for a specific time interval, up to 36 hours before the current time. Because the actual performance management objects might have changed since the time interval you request, or actually might not exist now, the performance management report APIs are not object-based.

Instead, all of the report APIs are implemented as performance-management specific operations that request an on-demand report to be generated and returned to the caller. They accept query parameters in a request block, and query the historical reporting database to generate a custom report for the caller.

The performance management report APIs are interrelated because you can use certain properties retrieved from one report as input to generate other reports, just as you can drill down for more information through the report tasks in the HMC UI.

Figure 222 on page 419 shows how the performance management reports are related, along with an indication of which properties you can use from one report to generate the next one through the APIs. Once you receive these properties in the response to an API request for one report, you can repeatedly reuse them as input properties for subsequent reports for the same performance management object, regardless of the time interval for the report.

For example, suppose that you use the API to request a Virtual Servers Report for the ensemble and the response body includes a **hypervisor-report-id** property. At any time in the future, you can use the **hypervisor-report-id** property directly to request a Hypervisor Report, rather than having to drill down to it again through the Virtual Servers Report. The ***-report-id** property values do not change across invocations of the reporting APIs; instead, the values remain constant over time.

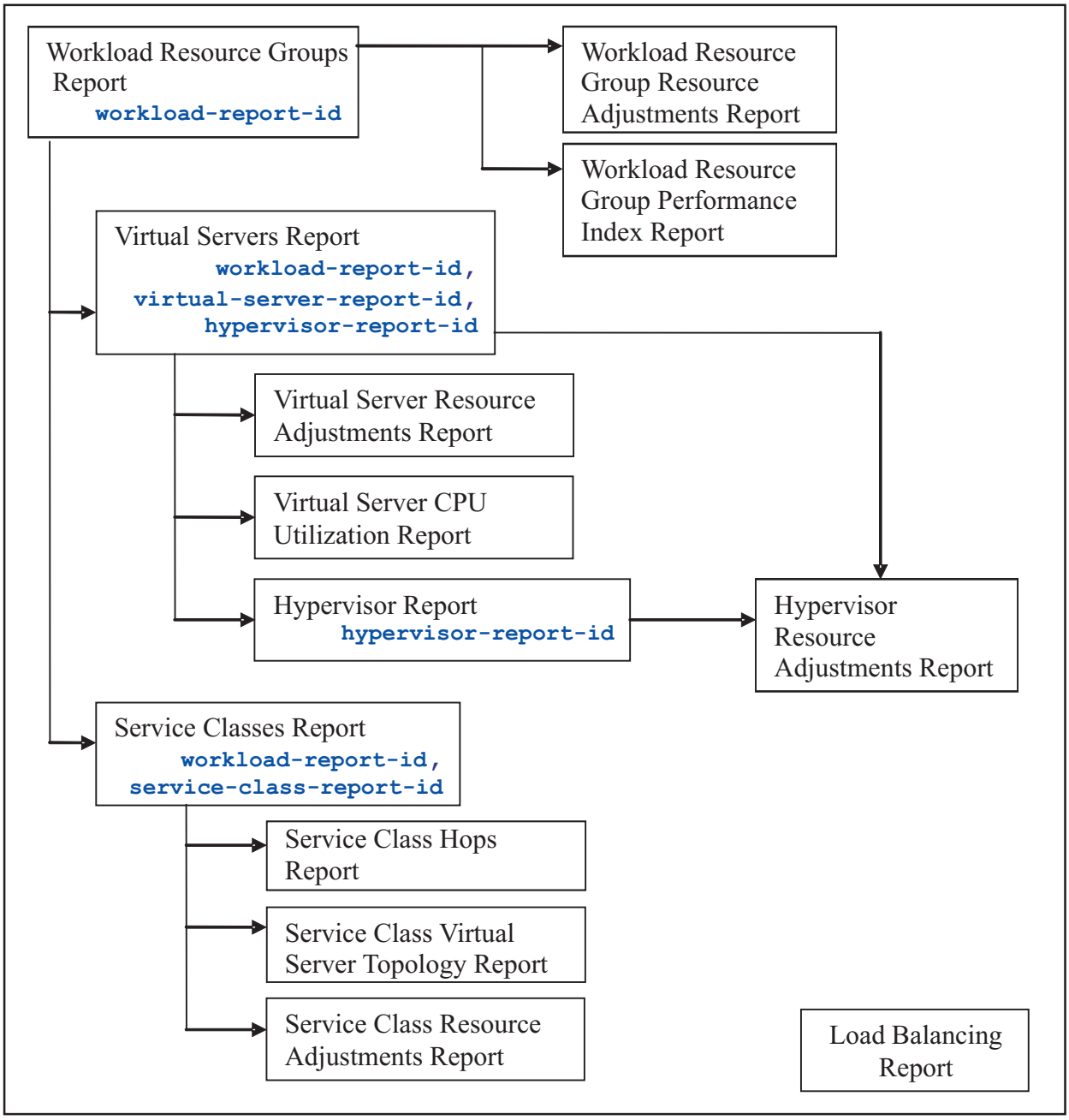

Figure 222. Relationship between reports and the properties used

Generate Workload Resource Groups Report

Use the **Generate Workload Resource Groups Report** operation to create a custom on-demand report that shows all workload resource groups in a specific ensemble over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

```
POST /api/ensembles/{ensemble-id}/performance-management/operations/generate-workload-resource
  -groups-report
```

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a workload resource groups report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`report-workloads`	Array of objects	Array of nested workload-report-entry objects described in Table 79.

Each nested workload-report-entry object contains the following fields:

Table 79. Format of a workload-report-entry object

Field name	Type	Description
`workload-name`	String	The displayable name of the workload resource group.
`workload-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.
`performance-policy-name`	String	The displayable name of the performance policy that was active during this reporting interval.

Table 79. Format of a workload-report-entry object (continued)

Field name	Type	Description
performance-policy-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific performance policy named in the **performance-policy-name** field. **Note:** If more than one performance policy was activated over the span of this reporting interval, only the name of the last one activated is returned in this field. The **multi-policy-activations** field indicates whether more than one policy was activated during this interval.
largest-pi-service-class-name	String	The displayable name of the performance management service class that had the largest PI (performance index) value over this reporting interval. **Optional:** This field is not returned if the performance index (PI) value could not be calculated.
largest-pi-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in **largest-pi-service-class-name** field. **Optional:** This field is not returned if the performance index (PI) value could not be calculated.
largest-pi	Number	The largest PI (performance index) value over this reporting interval. **Optional:** This field is not returned if the performance index (PI) value could not be calculated.
multi-policy-activations	Boolean	Indicates whether more than one performance policy was activated over the span of this reporting interval. The value is **"true"** if more than one policy or **"false"** if only one policy was activated during this time interval.
cpu-utilization-distribution	Array of objects	Array of nested cpu-utilization-range objects described in Table 80.
most-severe-perf-status	String Enum	The most severe **perf-status** value recorded for the workload over this reporting interval. See "Data model" on page 370 for more details about the valid values of this property.
perf-status-data-points	Array of objects	Array of nested perf-status-data-point objects described in Table 81.

Each nested cpu-utilization-range object contains the following fields:

Table 80. Format of a cpu-utilization-range object

Field name	Type	Description
low-boundary	Number	This value defines the low boundary of the CPU utilization range that this object is covering.
high-boundary	Number	This value defines the high boundary of the CPU utilization range that this object is covering.
virtual-server-count	Integer	This value is the number of virtual servers whose average CPU utilization was within the CPU utilization range during the reporting interval.

Each nested perf-status-data-point object contains the following fields:

Table 81. Format of a perf-status-data-point object

Field name	Type	Description
time	Integer	Standard date/time value indicating the date and time that this performance status value was recorded. The standard time value is defined as the number of elapsed milliseconds after midnight January 1, 1970.

Table 81. Format of a perf-status-data-point object (continued)

Field name	Type	Description
`perf-status`	Number	The **perf-status** value of the workload at the recorded time. See the "Data model" on page 370 for more details about the valid values of this property.

Description

The **Generate Workload Resource Groups Report** operation generates a report that contains the following information for the requested time interval:

- A list of all workload resource groups for which historical reporting information is available
- For each workload resource group:
  - The active performance policy at that time
  - A performance health indication (the performance index or PI) over that interval.

"Response body contents" on page 420 describes the full list of data that is returned in this report for each workload resource group. To request more detailed performance reporting data, you can use the returned **workload-report-id** property for a specific workload resource group as input for additional report operations.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 420. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Workloads Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 420.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-workload-resource-groups-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 30
}
```

Figure 223. Generate Workload Resource Groups Report: Request

Generate Workload Resource Group Performance Index Report

Use the **Generate Workload Resource Group Performance Index Report** operation to create a custom on-demand performance index report for a specific workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-workload-resource-group-performance-index-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a performance index report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
workload-report-id	String	Required	The identifier used by the performance management reporting to keep track of reporting data for the specific workload resource group for which the report is being requested. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.

Field name	Type	Description
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
workload-name	String	The displayable name of the workload resource group.
workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
pi-data-by-service-class	Array of objects	Array of nested service-class-pi-data objects described in Table 82.

Each nested service-class-pi-data object contains the following fields:

Table 82. Format of a service-class-pi-data object

Field name	Type	Description
service-class-name	String	The displayable name of the service class.
service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field.
pi-data-points	Array of objects	Array of nested pi-data-point objects described in Table 83.

Each nested pi-data-point object contains the following fields:

Table 83. Format of a pi-data-point object

Field name	Type	Description
pi-time	Integer	Standard date/time value indicating the requested starting date and time that this data point was taken. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
pi-value	Number	The specific performance index (PI) value recorded at the date/time in the **pi-time** field.

Description

The **Generate Workload Resource Group Performance Index Report** operation generates a report that contains the following information for a specific workload resource group over the requested time interval:

- A list of the services classes within the requested workload resource group for which performance index data is available
- For each services class, a list of all of the individual PI data points that were recorded over that interval. Each PI data point contains both the actual PI value and the date/time that it was recorded.

"Response body contents" on page 423 describes the full list of data that is returned in this report for each workload resource group. To request more detailed performance reporting data, you can use the returned **service-class-report-id** property for a specific service class as input for additional report operations.

Note that when you request performance index information for a workload resource group through the HMC UI, the resulting report is a line graph display that plots these PI values with plot points and a line over time for each service class.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 423. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Workloads Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 423.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-workload-resource-group-performance-index-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group"
}
```

Figure 224. Generate Workload Resource Group Performance Index Report: Request

Generate Workload Resource Group Resource Adjustments Report

Use the **Generate Workload Resource Group Resource Adjustments Report** operation to create a custom on-demand resource adjustments report for a specific workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-workload-resource-group-resource-adjustments-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a resource adjustments report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Req/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
workload-report-id	String	Required	The identifier used by the performance management reporting to keep track of reporting data for the specific workload resource group for which the report is being requested. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
workload-name	String	The displayable name of the workload resource group.
workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.

Field name	Type	Description
successful-resource-adjustments	Array of objects	Array of nested successful-resource-adjustment objects. If no successful adjustments occurred during the requested time interval, an empty array is returned.
failed-resource-adjustments	Array of objects	Array of nested failed-resource-adjustment objects. If no failed adjustments occurred during the requested time interval, an empty array is returned.

Each nested successful-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that was given additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that was given additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that was given additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
receiver-processing-units-after	Number	The total number of processing units the receiver had after the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
receiver-processing-units-before	Number	The total number of processing units the receiver had before the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
adjustment-donors	Array of objects	Array of nested adjustment-donors objects.

Each nested adjustment-donors object contains the following fields:

Field name	Type	Description
donor-virtual-server-name	String	The name of a virtual server that gave up resources as part of this adjustment.

Field name	Type	Description
donor-processing-units-after	Number	The total number of processing units this donor had after it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-processing-units-before	Number	The total number of processing units this donor had before it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-workload-names	Array of strings	The displayable names of all workload resource groups that donated resources when this donor virtual server gave up resources.

Each nested failed-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that needed but did not receive additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that needed but did not receive additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that needed but did not receive additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
adjustment-fail-reason	String Enum	The reason why the adjustment failed. Values are: • **"not-enough-capacity"** — The sum of the capacity that could be given up by all the available donors in the group was not enough to meet the capacity increase request of the intended receiver. • **"no-potential-donors"** — The group did not contain any potential donors so no assessments for increased capacity could be sent. • **"entitled-capacity-not-achievable"** — The extra capacity required to be added to the entitled capacity would have pushed the total beyond the maximum entitled capacity. • **"processor-not-fully-utilized"** — The virtual server is not fully using the processors it already has, so adding more will not have any effect. • **"requested-more-shares-than-max"** — More shares than the maximum allowed for this virtual server were requested. • **"not-enough-virtual-cpus"** — Not enough virtual CPUs were available for the projected required total consumed capacity to be achieved. • **"unknown"** — Unknown uncategorized failure reason.

Description

The **Generate Workload Resource Group Resource Adjustments Report** operation generates a report that contains the following information for a specific workload resource group over the requested time interval:

- A list of resource adjustments that zManager successfully made to maintain specified performance goals. For successful adjustments, the report includes:
  - A list of workload resource groups and the virtual servers that received additional resources (receivers)
  - A list of workload resource groups and the virtual servers that donated the additional resources (donors)
- A list of resource adjustments that zManager attempted but failed to make, along with a reason for the failure.

"Response body contents" on page 426 describes the full list of data that is returned in the report for this workload resource group.

Additional types of resource adjustment reports are available. This particular generate-workload-resource-group-resource-adjustments-report operation generates a resource adjustments report for a specific workload over the requested time period. What this means is that it will contain entries for all adjustments that affected the particular workload submitted; meaning all of the entries returned will involve that specific workload either having received additional resources, or having been forced to give up (donate) some of its resources.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 426. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Workloads Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 426.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-workload-resource-group-resource-adjustments-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group"
}
```

Figure 225. Generate Workload Resource Group Resource Adjustments Report: Request

Generate Virtual Servers Report

Use the **Generate Virtual Servers Report** operation to create a custom on-demand report that shows all virtual servers that were members of a particular workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-virtual-servers-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a virtual server report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Req/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
`workload-report-id`	String	Required	The identifier used by the performance management reporting to keep track of reporting data for the specific workload resource group for which the report is being requested. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`workload-name`	String	The displayable name of the workload resource group.
`workload-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
`report-virtual-servers`	Array of objects	Array of nested virtual-server-report-entry objects.

Each nested virtual-server-report-entry object contains the following fields:

Field name	Type	Description
`virtual-server-name`	String	The displayable name of the virtual server.
`virtual-server-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific virtual server named in the **virtual-server-name** field.
`hypervisor-type`	String	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
`hypervisor-name`	String	The displayable name of the hypervisor under which this virtual server was running.
`hypervisor-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique).
`defined-cpc-name`	String	The name of the central processor complex (CPC) with which this virtual server was associated.
`was-active`	Boolean	Indicates whether this virtual server was actually active during this reporting interval. It will be true if it was active and false if it was not active. If this field is false, none of the fields in this next table this one can have data; so in that case none of the fields below here will be returned.
`os-name`	String	The name of the operating system image that is running on the virtual server as known to its operating system. **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.

Field name	Type	Description
os-type	String	The type of operating system that is running on the virtual server. **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-level	String	The release level of the operating system that is running on the virtual server. **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
hostname-ipaddr	String	The host name (or IP address) of the virtual server. **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
virtual-processors	String	The number of virtual processors associated with this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
allocated-memory	String	For the x Hyp, PowerVM, and PR/SM hypervisor types, this field contains the allocated memory (in MB) configured for the virtual server. For z/VM, this field contains the amount of virtual memory currently resident in real memory for the guest. **Optional:** This field is returned only if the virtual server was active during this interval.
physical-cpu-utilization-percent	Number	The physical processor utilization percentage (%) of the virtual server. This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval.
hypervisor-cpu-delay-percent	Number	The hypervisor processing unit delay in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM and PowerVM. In the case of z/VM, a value is available for this field only if sampling is turned on for the guest. **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information.
idle-time-percent	Number	The idle time percentage (%) of the virtual server. It is the percentage of total time that the virtual server had no work (that is, no application processes or internal hypervisor specific process states). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information. For z/VM, this data is available only when sampling is enabled and started. For PowerVM and PR/SM, idle time data is not available.
other-time-percent	Number	The percentage (%) of total time that miscellaneous hypervisor specific internal process states had control for this virtual server. In other words, the percentage of time that the virtual server was not idle but also was not in a state of active CPU utilization or hypervisor CPU delay. This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information. For z/VM, this data is available only when sampling is enabled and started. For PowerVM and PR/SM, other time data is not available.
service-class-name	String	The displayable name of the service class. **Optional:** This field is returned only if the virtual server was active and if the virtual server was associated with a specific service class within the requested workload resource group during this interval.
service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field. **Optional:** This field is returned only if the **service-class-name** field is returned.

Field name	Type	Description
pi-value	Number	The average performance index (PI) value calculated over this reporting interval for the service class returned in the **service-class-name** field. This field is not returned if the virtual server was not active or if a PI value could not be calculated.
os-cpu-using-samples-percent	Number	The percentage of CPU using samples from among the total samples. For example, if there are 10 CPU using samples out of a total of 10 samples, then CPU using samples is 100% (because out of the total samples, all are CPU using samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-cpu-delay-samples-percent	Number	The percentage of CPU delay samples from among the total samples. For example, if there are 10 CPU delay samples and 10 samples that are not CPU delay samples, then CPU delay samples is 50% (because out of the total samples half are CPU delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-io-delay-samples-percent	Number	The percentage of I/O delay samples from among the total samples. The percent I/O delay is the percent of samples taken when work was delayed for non-paging DASD I/O. The I/O delay includes IOS queue, subchannel pending, and control unit queue delays. For example, if there are 10 I/O delay samples and 10 samples that are not I/O delay samples, then I/O delay samples is 50% (because out of the total samples half are I/O delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-page-delay-samples-percent	Number	The percentage of page delay samples from among the total samples. The percent page delay is the percent of samples when the address space experienced page faults in cross-memory access, and the page faults were resolved from auxiliary storage. For example, if there are 10 page delay samples and 10 samples that are not page delay samples, then page delay samples is 50% (because out of the total samples half are page delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.

Description

The **Generate Virtual Servers Report** operation generates a report that contains the following information for a specific workload resource group over the requested time interval:

- A list of all virtual servers that were members in that specific workload resource group for which historical reporting information is available
- For each virtual server:
  - The unique performance reporting identifier for the virtual server
  - The name and unique performance reporting identifier of the hypervisor under which the virtual server was running
  - An indication of whether the virtual server was active over the reporting interval.

"Response body contents" on page 431 describes the full list of data that is returned in this report for each virtual server. To request more detailed performance reporting data, you can use the returned **virtual-server-report-id**, **hypervisor-report-id**, and **service-class-report-id** properties as input for additional report operations.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 430. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Virtual Servers Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 431.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-virtual-servers-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group"
}
```

Figure 226. Generate Virtual Servers Report: Request

Generate Virtual Server CPU Utilization Report

Use the **Generate Virtual Server CPU Utilization Report** operation to create a custom on-demand report that shows the processor (CPU) utilization for a specific virtual server over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-virtual-server-cpu-utilization-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a virtual server CPU utilization report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
`virtual-server-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific virtual server named in the **virtual-server-name** field. This value is the same as one of the **virtual-server-report-id** values that are returned in the Virtual Servers Report. Alternatively, you can supply the **object-id** property value of an existing `virtual server` object.
`hypervisor-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique). This value is the same as the **hypervisor-report-id** value that is returned for this virtual server in the Virtual Servers Report. Alternatively, you can supply the **object-id** property value of an existing `virtualization host` object.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`virtual-server-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific virtual server named in the **virtual-server-name** field.
`virtual-server-name`	String	The displayable name of the virtual server.

Field name	Type	Description
hypervisor-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique).
hypervisor-name	String	The displayable name of the hypervisor under which this virtual server was running.
cpu-utilization-data-points	Array of objects	Array of nested cpu-utilization-data-points objects.

Each nested cpu-utilization-data-points object contains the following fields:

Field name	Type	Description
cpu-utilization-time	Integer	Standard date/time value indicating the requested starting date and time that this data point was taken. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
cpu-utilization-value	Number	The specific CPU utilization value recorded at the date/time in the **cpu-utilization-time** field. This value is in fractional form representing a percentage (between 0 and 1 inclusive).

Description

The **Generate Virtual Server CPU Utilization Report** operation generates a report that contains the following information for a specific virtual server over the requested time interval:

- All of the individual CPU utilization data points for that virtual server that were recorded over the requested reporting interval. Each CPU utilization data point contains both the actual CPU utilization value and the date/time that it was recorded.

"Response body contents" on page 435 describes the full list of data that is returned in this report for each virtual server.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 435. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Virtual Servers Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 435.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-virtual-server-cpu-utilization-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "virtual-server-report-id": "vs241510-5678900000-xxxxx",
 "hypervisor-report-id": "phyp241510-5678900000-xxxxx"
}
```

Figure 227. Generate Virtual Server CPU Utilization Report: Request

Generate Virtual Server Resource Adjustments Report

Use the **Generate Virtual Server Resource Adjustments Report** operation to create a custom on-demand resource adjustments report for a specific virtual server over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-virtual-server-resource-adjustments-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a virtual server resource adjustments report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.

Field name	Type	Rqd/Opt	Description
`workload-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.
`virtual-server-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific virtual server named in the **virtual-server-name** field. This value is the same as one of the **virtual-server-report-id** values that are returned in the Virtual Servers Report. Alternatively, you can supply the **object-id** property value of an existing `virtual server` object.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`workload-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
`virtual-server-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific virtual server named in the **virtual-server-name** field.
`successful-resource-adjustments`	Array of objects	Array of nested `successful-resource-adjustment` objects. If no successful adjustments occurred during the requested time interval, an empty array is returned.
`failed-resource-adjustments`	Array of objects	Array of nested `failed-resource-adjustment` objects. If no failed adjustments occurred during the requested time interval, an empty array is returned.

Each nested `successful-resource-adjustment` object contains the following fields:

Field name	Type	Description
`hypervisor-type`	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
`adjustment-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.

Field name	Type	Description
receiver-virtual-server-name	String	The name of the virtual server that was given additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that was given additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that was given additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
receiver-processing-units-after	Number	The total number of processing units the receiver had after the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
receiver-processing-units-before	Number	The total number of processing units the receiver had before the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
adjustment-donors	Array of objects	Array of nested adjustment-donors objects.

Each nested adjustment-donors object contains the following fields:

Field name	Type	Description
donor-virtual-server-name	String	The name of a virtual server that gave up resources as part of this adjustment.
donor-processing-units-after	Number	The total number of processing units this donor had after it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-processing-units-before	Number	The total number of processing units this donor had before it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-workload-names	Array of strings	The displayable names of all workload resource groups that donated resources when this donor virtual server gave up resources.

Each nested failed-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that needed but did not receive additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that needed but did not receive additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that needed but did not receive additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
adjustment-fail-reason	String Enum	The reason why the adjustment failed. Values are: • **"not-enough-capacity"** — The sum of the capacity that could be given up by all the available donors in the group was not enough to meet the capacity increase request of the intended receiver. • **"no-potential-donors"** — The group did not contain any potential donors so no assessments for increased capacity could be sent. • **"entitled-capacity-not-achievable"** — The extra capacity required to be added to the entitled capacity would have pushed the total beyond the maximum entitled capacity. • **"processor-not-fully-utilized"** — The virtual server is not fully using the processors it already has, so adding more will not have any effect. • **"requested-more-shares-than-max"** — More shares than the maximum allowed for this virtual server were requested. • **"not-enough-virtual-cpus"** — Not enough virtual CPUs were available for the projected required total consumed capacity to be achieved. • **"unknown"** — Unknown uncategorized failure reason.

Description

The **Generate Virtual Server Resource Adjustments Report** operation generates a report that contains the following information for a specific virtual server over the requested time interval:

- A list of resource adjustments that zManager successfully made to maintain specified performance goals. For successful adjustments, the report includes:
  - A list of workload resource groups and the virtual servers that received additional resources (receivers)
  - A list of workload resource groups and the virtual servers that donated the additional resources (donors)
- A list of resource adjustments that zManager attempted but failed to make, along with a reason for the failure.

"Response body contents" on page 438 describes the full list of data that is returned in the report for this workload resource group.

Additional types of resource adjustment reports are available. This particular generate-virtual-server-resource-adjustments-report operation generates a resource adjustments report for a specific virtual server within a specific workload over the requested time period. What this means is that it will contain entries

for all adjustments that affected the particular virtual server submitted; meaning all of the entries returned will involve that specific virtual server either having received additional resources, or having been forced to give up (donate) some of its resources.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 437. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Virtual Server Resource Adjustments Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 438.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-virtual-server-resource-adjustments-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group",
 "virtual-server-report-id": "vs241510-5678900000-xxxxx"
}
```

Figure 228. Generate Virtual Server Resource Adjustments Report: Request

Generate Hypervisor Report

Use the **Generate Hypervisor Report** operation to create a custom on-demand report that shows information about a specific hypervisor and all virtual servers that were running under that hypervisor over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-hypervisor-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a hypervisor report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
hypervisor-report-id	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique).

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
hypervisor-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique).
hypervisor-name	String	The displayable name of the hypervisor under which this virtual server was running.

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
report-hypervisor-details	Object	An object that contains report details for the hypervisor itself. The format of the returned object is described in Table 84.
report-hypervisor-virtual-servers	Array of objects	An array of nested objects, each representing a virtual server that was under the control of this hypervisor during this reporting interval. The format of the returned object depends on the value returned in the hypervisor-type field: • For **"power-vm"**, see Table 85 on page 444 • For **"x-hyp"**, see Table 86 on page 445 • For **"zvm"**, see Table 87 on page 446 • For **"prsm"**, see Table 88 on page 447
resource-adjustment-report	Object	If a hypervisor-level resource adjustment report is available for this same reporting interval, this object is returned. The format of the resource-adjustment-report object is the same as that described in "Response body contents" on page 450 for the **Generate Hypervisor Resource Adjustments Report** operation.

The report-hypervisor-details object for a hypervisor contains the fields in Table 84.

Table 84. Format of a report-hypervisor-details object

Field name	Type	Description
allocated-processing-units	Number	The total number of processing units that were allocated. This field is included in the object only for the PowerVM hypervisor type.
processor-count	Integer	The total number of physical processors.
cpu-consumption-percent	Number	The total consumption (utilization) percentage (%) of hypervisor CPUs. This percentage is in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM, x Hyp, and PowerVM.
shared-cp-consumption-percent	Number	The total consumption (utilization) percentage (%) of shared processors. This percentage is in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisor only: PR/SM.
memory-used	Integer	The total amount of the hypervisor memory that was in use (in MB).
total-memory	Integer	The total amount of memory (in MB) that was that was available to the hypervisor.
cp-cpu-consumption-percent	Number	The total consumption (utilization) percentage (%) of the hypervisor's general purpose CPUs. It is in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisor only: z/VM.
ifl-cpu-consumption-percent	Number	The total consumption (utilization) percentage (%) of the hypervisor's IFL CPUs. It is in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisor only: z/VM.
other-cpu-consumption-percent	Number	The total consumption (utilization) percentage (%) of the hypervisor's CPU types other than CPs and IFLs. It is in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisor only: z/VM.

Each nested report-hypervisor-virtual-servers object for a PowerVM hypervisor contains the fields in Table 85 on page 444.

Table 85. Format of a PowerVM report-hypervisor-virtual-servers object

Field name	Type	Description
`virtual-server-name`	String	The displayable name of the virtual server.
`workload-processor-mgmt-status`	String Enum	The processor management status, which is one of the following values: • **"active"** • **"not-active"**
`workload-processor-mgmt-status-reason`	String Enum	A further explanation of the reason for processor management status returned in the **workload-processor-mgmt-status** field. Values are: • **"none"**– the value of **workload-processor-mgmt-status** is **"active"** • **"mgmt-disabled-global"**– the option to enable processor performance management at the hypervisor type level was not set for this hypervisor type • **"mgmt-disabled-vs"**– the option to enable processor performance management at the virtual server level was not set for this virtual server • **"dedicated-proc-mode"**– the virtual server is running in dedicated processor mode, and only shared mode is supported for processor management • **"internal-error"**– An internal error has occurred. • **"network-connection-failure"**– the virtual server has no connectivity to its hypervisor. • **"virtual-server-not-active"**– the virtual server itself was not up and active during this reporting interval
`was-active`	Boolean	Indicates whether this virtual server was active during this reporting interval. The value is **"true"** if the virtual server was active or **"false"** if it was not active. If the value for this field is false, none of the remaining fields in this object can have data so they are not returned.
`virtual-processor-count`	Integer	This value is the number of virtual processors that were associated with this virtual server during the reporting interval. **Optional:** This field is returned only if the virtual server was active during this interval.
`min-virtual-processors`	Integer	The minimum number of virtual processors allowed for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
`max-virtual-processors`	Integer	The maximum number of virtual processors allowed for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
`consumed-processors`	Number	The consumed processors metric for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
`was-dedicated`	Boolean	Indicates whether this virtual server was running in dedicated mode (as opposed to shared mode) during this reporting interval. The value is **"true"** if the virtual server was in dedicated mode or **"false"** if it was in shared mode. **Optional:** This field is returned only if the virtual server was active during this interval.
`was-capped`	Boolean	Indicates whether this virtual server was capped during this reporting interval. The value is **"true"** if the virtual server was capped or **"false"** if it was not capped. **Optional:** This field is returned only if the virtual server was active during this interval.
`hypervisor-cpu-delay-percent`	Number	The hypervisor processing unit delay in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM and PowerVM. In the case of z/VM, a value is available for this field only if sampling is turned on for the guest. This field is returned only if the virtual server was active and running in shared mode (as opposed to dedicated mode) during this interval.

Table 85. Format of a PowerVM report-hypervisor-virtual-servers object (continued)

Field name	Type	Description
processing-units	Number	The number of processing units that were assigned to this virtual server during the reporting interval. **Optional:** This field is returned only if the virtual server was active during this interval.
initial-processing-units	Number	The number of processing units that were initially assigned to this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
min-processing-units	Number	The minimum number of processing units that this virtual server could use. **Optional:** This field is returned only if the virtual server was active during this interval.
max-processing-units	Number	The maximum number of processing units that this virtual server could use. **Optional:** This field is returned only if the virtual server was active during this interval.
min-memory	Integer	The minimum amount of memory (in MB) that this virtual server could use. **Optional:** This field is returned only if the virtual server was active during this interval.
max-memory	Integer	The maximum amount of memory (in MB) that this virtual server could use. **Optional:** This field is returned only if the virtual server was active during this interval.

Each nested report-hypervisor-virtual-servers object for an x Hyp hypervisor contains the fields in Table 86.

Table 86. Format of an x Hyp report-hypervisor-virtual-servers object

Field name	Type	Description
virtual-server-name	String	The displayable name of the virtual server.
was-active	Boolean	Indicates whether this virtual server was active during this reporting interval. The value is **"true"** if the virtual server was active or **"false"** if it was not active. If the value for this field is false, none of the remaining fields in this object can have data so they are not returned.
virtual-processor-count	Integer	This value is the number of virtual processors that were associated with this virtual server during the reporting interval. **Optional:** This field is returned only if the virtual server was active during this interval.
consumed-processors	Number	The consumed processors metric for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
hypervisor-cpu-delay-percent	Number	The hypervisor processing unit delay in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM and PowerVM. In the case of z/VM, a value is available for this field only if sampling is turned on for the guest. **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
memory-in-use	Integer	The amount of memory (in MB) that was actually in use by this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
allocated-memory	Integer	For the x Hyp, PowerVM, and PR/SM hypervisor types, this field contains the allocated memory (in MB) configured for the virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.

Each nested report-hypervisor-virtual-servers object for a z/VM hypervisor contains the fields in Table 87.

Table 87. Format of a z/VM report-hypervisor-virtual-servers object

Field name	Type	Description
`virtual-server-name`	String	The displayable name of the virtual server.
`workload-processor-mgmt-status`	String Enum	The processor management status, which is one of the following values: • **"active"** • **"not-active"**
`workload-processor-mgmt-status-reason`	String Enum	A further explanation of the reason for processor management status returned in the **workload-processor-mgmt-status** field. Values are: • **"none"**– the value of **workload-processor-mgmt-status** is **"active"** • **"mgmt-disabled-global"**– the option to enable processor performance management at the hypervisor type level was not set for this hypervisor type • **"mgmt-disabled-vs"**– the option to enable processor performance management at the virtual server level was not set for this virtual server • **"absolute-share-mode"**– the virtual server is running in absolute share mode, and only relative share mode is supported for processor management • **"sampling-disabled"**– z/VM sampling was not enabled. Sampling must be enabled for processor management under z/VM. • **"internal-error"**– An internal error has occurred. • **"network-connection-failure"**– the virtual server has no connectivity to its hypervisor. • **"virtual-server-not-active"**– the virtual server itself was not up and active during this reporting interval
`was-active`	Boolean	Indicates whether this virtual server was active during this reporting interval. The value is **"true"** if the virtual server was active or **"false"** if it was not active. If the value for this field is false, none of the remaining fields in this object can have data so they are not returned.
`virtual-processor-count`	Integer	This value is the number of virtual processors that were associated with this virtual server during the reporting interval. **Optional:** This field is returned only if the virtual server was active during this interval.
`consumed-processors`	Number	The consumed processors metric for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
`hypervisor-cpu-delay-percent`	Number	The hypervisor processing unit delay in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM and PowerVM. In the case of z/VM, a value is available for this field only if sampling is turned on for the guest. This field is returned only if the virtual server was active and sampling is enabled during this interval.
`share-mode`	String Enum	The processor share mode configured for this virtual server during this reporting interval. Valid values are **"absolute"** and **"relative"**. **Optional:** This field is returned only if the virtual server was active during this interval.
`share-limit`	String Enum	The processor share limit configured for this virtual server during this reporting interval. Valid values are **"soft"**, **"hard"** and **"none"**. **Optional:** This field is returned only if the virtual server was active during this interval.
`shares`	Integer	The number of processor shares that were associated with this virtual server during this reporting interval. **Optional:** This field is returned only if the virtual server was active and running in relative share mode (as opposed to absolute share mode) during this interval.

Table 87. Format of a z/VM report-hypervisor-virtual-servers object (continued)

Field name	Type	Description
min-shares	Integer	The minimum number of processor shares that could be dynamically assigned to the virtual server during this reporting interval. **Optional:** This field is returned only if the virtual server was active and running in relative share mode (as opposed to absolute share mode) during this interval.
max-shares	Integer	The maximum number of processor shares that could be dynamically assigned to the virtual server during this reporting interval. **Optional:** This field is returned only if the virtual server was active and running in relative share mode (as opposed to absolute share mode) during this interval.
memory-used	Integer	The total amount of the hypervisor memory that was in use (in MB). **Optional:** This field is returned only if the virtual server was active during this interval.

Each nested report-hypervisor-virtual-servers object for a PR/SM hypervisor contains the fields in Table 88.

Table 88. Format of a PR/SM report-hypervisor-virtual-servers object

Field name	Type	Description
virtual-server-name	String	The displayable name of the virtual server.
was-active	Boolean	Indicates whether this virtual server was active during this reporting interval. The value is **"true"** if the virtual server was active or **"false"** if it was not active. If the value for this field is false, none of the remaining fields in this object can have data so they are not returned.
logical-processor-count	Integer	The number of logical processors that were associated with this virtual server during this reporting interval. **Optional:** This field is returned only if the virtual server was active during this interval.
consumed-processors	Number	The consumed processors metric for this virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
allocated-memory	Integer	For the x Hyp, PowerVM, and PR/SM hypervisor types, this field contains the allocated memory (in MB) configured for the virtual server. **Optional:** This field is returned only if the virtual server was active during this interval.
was-dedicated	Boolean	Indicates whether this virtual server was running in dedicated mode (as opposed to shared mode) during this reporting interval. The value is **"true"** if the virtual server was in dedicated mode or **"false"** if it was in shared mode. **Optional:** This field is returned only if the virtual server was active during this interval.
cpu-weight	Integer	The CPU weight that was associated to this virtual server during this reporting interval. This field is returned only if the virtual server was active and running in shared mode (as opposed to dedicated mode) during this interval.
min-cpu-weight	Integer	The minimum CPU weight that this virtual server could use. This field is returned only if the virtual server was active and running in shared mode (as opposed to dedicated mode) during this interval.
max-cpu-weight	Integer	The maximum CPU weight that this virtual server could use. This field is returned only if the virtual server was active and running in shared mode (as opposed to dedicated mode) during this interval.

Description

The **Generate Hypervisor Report** operation generates a report that contains the following information for a specific hypervisor over the requested time interval:
- Information about the hypervisor itself, including the total number of physical processors and total consumption percentage of those processors
- A list of all virtual servers that were running under this specific hypervisor for which historical reporting information is available
- For each virtual server:
  - An indication of whether the virtual server was active over the reporting interval
  - Additional information such as processor counts and memory statistics.

"Response body contents" on page 442 describes the full list of data that is returned in this report for each virtual server. The information available for virtual servers varies depending on the hypervisor type.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 442. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Hypervisor Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 442.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-hypervisor-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "hypervisor-report-id": "phyp241510-5678900000-xxxxx"
}
```

Figure 229. Generate Hypervisor Report: Request

Generate Hypervisor Resource Adjustments Report

Use the **Generate Hypervisor Resource Adjustments Report** operation to create a custom on-demand resource adjustments report for a specific hypervisor over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-hypervisor-resource-adjustments-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a hypervisor resource adjustments report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
`hypervisor-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique). This value is the same as the **hypervisor-report-id** value that is returned for this virtual server in the Virtual Servers Report. Alternatively, you can supply the **object-id** property value of an existing `virtualization host` object.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`hypervisor-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific hypervisor under which this virtual server was running. (Hypervisor names returned in the **hypervisor-name** field are not unique).
`successful-resource-adjustments`	Array of objects	Array of nested `successful-resource-adjustment` objects. If no successful adjustments occurred during the requested time interval, an empty array is returned.
`failed-resource-adjustments`	Array of objects	Array of nested `failed-resource-adjustment` objects. If no failed adjustments occurred during the requested time interval, an empty array is returned.

Each nested `successful-resource-adjustment` object contains the following fields:

Field name	Type	Description
`hypervisor-type`	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
`adjustment-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`receiver-virtual-server-name`	String	The name of the virtual server that was given additional resources.
`receiver-workload-name`	String	The displayable name of the workload resource group that was given additional resources.
`receiver-workload-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
`receiver-service-class-name`	String	The displayable name of the service class within the workload resource group that was given additional resources.
`receiver-service-class-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.

Field name	Type	Description
receiver-processing-units-after	Number	The total number of processing units the receiver had after the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
receiver-processing-units-before	Number	The total number of processing units the receiver had before the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
adjustment-donors	Array of objects	Array of nested adjustment-donors objects.

Each nested adjustment-donors object contains the following fields:

Field name	Type	Description
donor-virtual-server-name	String	The name of a virtual server that gave up resources as part of this adjustment.
donor-processing-units-after	Number	The total number of processing units this donor had after it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-processing-units-before	Number	The total number of processing units this donor had before it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-workload-names	Array of strings	The displayable names of all workload resource groups that donated resources when this donor virtual server gave up resources.

Each nested failed-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that needed but did not receive additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that needed but did not receive additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.

Field name	Type	Description
`receiver-service-class-name`	String	The displayable name of the service class within the workload resource group that needed but did not receive additional resources.
`receiver-service-class-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
`adjustment-fail-reason`	String Enum	The reason why the adjustment failed. Values are: • **"not-enough-capacity"** — The sum of the capacity that could be given up by all the available donors in the group was not enough to meet the capacity increase request of the intended receiver. • **"no-potential-donors"** — The group did not contain any potential donors so no assessments for increased capacity could be sent. • **"entitled-capacity-not-achievable"** — The extra capacity required to be added to the entitled capacity would have pushed the total beyond the maximum entitled capacity. • **"processor-not-fully-utilized"** — The virtual server is not fully using the processors it already has, so adding more will not have any effect. • **"requested-more-shares-than-max"** — More shares than the maximum allowed for this virtual server were requested. • **"not-enough-virtual-cpus"** — Not enough virtual CPUs were available for the projected required total consumed capacity to be achieved. • **"unknown"** — Unknown uncategorized failure reason.

Description

The **Generate Hypervisor Resource Adjustments Report** operation generates a report that contains the following information for a specific hypervisor over the requested time interval:

- A list of resource adjustments that zManager successfully made to maintain specified performance goals. For successful adjustments, the report includes:
  - A list of workload resource groups and the virtual servers that received additional resources (receivers)
  - A list of workload resource groups and the virtual servers that donated the additional resources (donors)
- A list of resource adjustments that zManager attempted but failed to make, along with a reason for the failure.

"Response body contents" on page 450 describes the full list of data that is returned in the report for this workload resource group.

Additional types of resource adjustment reports are available. This particular generate-hypervisor-resource-adjustments-report operation generates a resource adjustments report for a specific hypervisor over the requested time period. What this means is that it will contain entries for all adjustments that affected virtual servers within the particular hypervisor submitted; meaning all of the entries returned will involve a virtual server under the control of that specific hypervisor either having received additional resources, or having been forced to give up (donate) some of its resources.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements" on page 453.

The request body is validated against the schema described in "Request body contents" on page 449. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Hypervisor Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 450.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-hypervisor-resource-adjustments-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "hypervisor-report-id": "phyp241510-5678900000-xxxxx"
}
```

Figure 230. Generate Hypervisor Resource Adjustments Report: Request

Generate Service Classes Report

Use the **Generate Service Classes Report** operation to create a custom on-demand report that shows all service classes within a particular workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-service-classes-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a service classes report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Req/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
workload-report-id	String	Required	The identifier used by the performance management reporting to keep track of reporting data for the specific workload resource group for which the report is being requested. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
workload-name	String	The displayable name of the workload resource group.
workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
performance-policy-name	String	The displayable name of the performance policy that contains this service class.
performance-policy-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific performance policy named in the **performance-policy-name** field.
report-service-classes	Array of objects	Array of nested service-class-report-entry objects.

Each nested service-class-report-entry object contains the following fields:

Field name	Type	Description
service-class-name	String	The displayable name of the service class. **Optional:** This field is returned only if the virtual server was active and if the virtual server was associated with a specific service class within the requested workload resource group during this interval.
service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field. **Optional:** This field is returned only if the **service-class-name** field is returned.
has-hop-data	Boolean	Indicates whether this service class has hop level data available for this reporting interval. The value is **"true"** if hop data is available or **"false"** if it is not available.
goal-type	String Enum	The type of performance goal that was defined for this service class. Values are **"velocity"** or **"discretionary"**.
goal-performance-level	String Enum	The performance goal of the service class as defined in performance policy. Possible values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**. This field is returned only if a velocity performance goal was defined for this service class.
business-importance	String Enum	The business importance level that was defined for this service class. Possible values are: **"highest"**, **"high"**, **"medium"**, **"low"**, and **"lowest"**. This field is returned only if a velocity performance goal was defined for this service class.
pi-value	Number	The average performance index (PI) value calculated over this reporting interval for the service class returned in the **service-class-name** field. This field is not returned if the virtual server was not active or if a PI value could not be calculated.
actual-performance-level	String Enum	The average level of performance that was actually measured over this reporting interval. Possible values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**. This field is returned only if a velocity performance goal was defined for this service class and if the performance index value could be measured.

Description

The **Generate Service Classes Report** operation generates a report that contains the following information for a specific workload resource group over the requested time interval:

- A list of all service classes within that specific workload resource group for which historical reporting information is available
- For each service class:
  - The name and unique reporting ID of the performance policy that contains the service class
  - An indication of whether the service class has available hop data for this reporting interval
  - Details about the service class definition, such as the type of performance goal
  - The actual performance level achieved during this reporting interval.

"Response body contents" on page 454 describes the full list of data that is returned in this report for each service class. If hop data is available for this service class (check the **has-hop-data** field), you can use the **service-class-report-id** property as input for additional report operations related to service classes:

- "Generate Service Class Hops Report" on page 461
- "Generate Service Class Virtual Server Topology Report" on page 466

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements" on page 456.

The request body is validated against the schema described in "Request body contents" on page 454. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Service Classes Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 454.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-service-classes-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group"
}
```

Figure 231. Generate Service Classes Report: Request

Generate Service Class Resource Adjustments Report

Use the **Generate Service Class Resource Adjustments Report** operation to create a custom on-demand resource adjustments report for a specific service class within a workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/
generate-service-class-resource-adjustments-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a service class resource adjustments report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`report-interval-start-time`	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
`report-interval-duration`	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
`workload-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.
`service-class-report-id`	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field. **Note:** This value is the same as the **service-class-report-id** value that is returned in one of the service class entries in the Service Classes Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`report-interval-start-time`	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
`report-interval-duration`	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
`workload-name`	String	The displayable name of the workload resource group.
`workload-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
`service-class-name`	String	The displayable name of the service class.
`service-class-report-id`	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field.

Field name	Type	Description
successful-resource-adjustments	Array of objects	Array of nested successful-resource-adjustment objects. If no successful adjustments occurred during the requested time interval, an empty array is returned.
failed-resource-adjustments	Array of objects	Array of nested failed-resource-adjustment objects. If no failed adjustments occurred during the requested time interval, an empty array is returned.

Each nested successful-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that was given additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that was given additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that was given additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
receiver-processing-units-after	Number	The total number of processing units the receiver had after the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
receiver-processing-units-before	Number	The total number of processing units the receiver had before the additional resources were given to it. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
adjustment-donors	Array of objects	Array of nested adjustment-donors objects.

Each nested adjustment-donors object contains the following fields:

Field name	Type	Description
donor-virtual-server-name	String	The name of a virtual server that gave up resources as part of this adjustment.

Field name	Type	Description
donor-processing-units-after	Number	The total number of processing units this donor had after it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-processing-units-before	Number	The total number of processing units this donor had before it gave up resources. Processing units are returned in the units that are normal for the specific hypervisor type of the virtual server. The units are entitled capacity for PowerVM, and CPU shares for all other hypervisor types.
donor-workload-names	Array of strings	The displayable names of all workload resource groups that donated resources when this donor virtual server gave up resources.

Each nested failed-resource-adjustment object contains the following fields:

Field name	Type	Description
hypervisor-type	String Enum	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
adjustment-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
receiver-virtual-server-name	String	The name of the virtual server that needed but did not receive additional resources.
receiver-workload-name	String	The displayable name of the workload resource group that needed but did not receive additional resources.
receiver-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **receiver-workload-name** field.
receiver-service-class-name	String	The displayable name of the service class within the workload resource group that needed but did not receive additional resources.
receiver-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **receiver-service-class-name** field.
adjustment-fail-reason	String Enum	The reason why the adjustment failed. Values are: • **"not-enough-capacity"** — The sum of the capacity that could be given up by all the available donors in the group was not enough to meet the capacity increase request of the intended receiver. • **"no-potential-donors"** — The group did not contain any potential donors so no assessments for increased capacity could be sent. • **"entitled-capacity-not-achievable"** — The extra capacity required to be added to the entitled capacity would have pushed the total beyond the maximum entitled capacity. • **"processor-not-fully-utilized"** — The virtual server is not fully using the processors it already has, so adding more will not have any effect. • **"requested-more-shares-than-max"** — More shares than the maximum allowed for this virtual server were requested. • **"not-enough-virtual-cpus"** — Not enough virtual CPUs were available for the projected required total consumed capacity to be achieved. • **"unknown"** — Unknown uncategorized failure reason.

Description

The **Generate Service Class Resource Adjustments Report** operation generates a report that contains the following information for a specific service class within a workload resource group over the requested time interval:

- A list of resource adjustments that zManager successfully made to maintain specified performance goals. For successful adjustments, the report includes:
  - A list of workload resource groups and the virtual servers that received additional resources (receivers)
  - A list of workload resource groups and the virtual servers that donated the additional resources (donors)
- A list of resource adjustments that zManager attempted but failed to make, along with a reason for the failure.

"Response body contents" on page 457 describes the full list of data that is returned in the report for this workload resource group.

There are multiple types of resource adjustments reports that can be requested, and each has its own API listed in this Performance Management Reports section. This particular generate-service-class-resource-adjustments-report operation generates a resource adjustments report for a specific service class within a specific workload over the requested time period. What this means is that it will contain entries for all adjustments that affected the particular service class submitted; meaning all of the entries returned will involve that specific service class either having received additional resources, or having been forced to give up (donate) some of its resources.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 457. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:

- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Service Class Resource Adjustments Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 457.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-service-class-resource-adjustments-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group",
 "service-class-report-id": "Batch Service"
}
```

Figure 232. Generate Service Class Resource Adjustments Report: Request

Generate Service Class Hops Report

Use the **Generate Service Class Hops Report** operation to create a custom on-demand hops report for a service class within a particular workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

Typically, a transactional service class has multiple hops, each hop corresponding to a tier in the transactional flow. Each hop can have one or more application environments associated with it; an application environment includes software and the server or network infrastructure that supports it. An application environment, in turn, might have multiple application environment servers.

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-service-class-hops-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object for which you are requesting a hops report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
workload-report-id	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.

Field name	Type	Rqd/Opt	Description
service-class-report-id	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field. **Note:** This value is the same as the **service-class-report-id** value that is returned in one of the service class entries in the Service Classes Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
workload-name	String	The displayable name of the workload resource group.
workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
service-class-name	String	The displayable name of the service class.
service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field.
performance-policy-name	String	The displayable name of the performance policy that contains this service class.
performance-policy-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific performance policy named in the **performance-policy-name** field.
goal-type	String Enum	The type of performance goal that was defined for this service class. Values are **"velocity"** or **"discretionary"**.
goal-performance-level	String Enum	The performance goal of the service class as defined in performance policy. Possible values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**. This field is returned only if a velocity performance goal was defined for this service class.
business-importance	String Enum	The business importance level that was defined for this service class. Possible values are: **"highest"**, **"high"**, **"medium"**, **"low"**, and **"lowest"**. This field is returned only if a velocity performance goal was defined for this service class.
pi-value	Number	The average performance index (PI) value calculated over this reporting interval for the service class returned in the **service-class-name** field. This field is not returned if the virtual server was not active or if a PI value could not be calculated.

Field name	Type	Description
actual-performance-level	String Enum	The average level of performance that was actually measured over this reporting interval. Possible values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**. This field is returned only if a velocity performance goal was defined for this service class and if the performance index value could be measured.
equivalent-workload-service-classes	Array of objects	Because a specific virtual server can belong to more than one workload, the hops report might contain reflect combined values for multiple workload resource groups rather than those for the requested workload resource group and service class. In this case, this field contains an array of nested equivalent-workload-service-class objects, each of which identifies another workload resource group or service class, if any, that have virtual servers also contained within the reported hops. The format of the returned object is described in Table 89. Otherwise, this field contains an empty array.
hops	Array of objects	An array of nested hop-entry objects. The format of the returned object is described in Table 90.

Each nested equivalent-workload-service-class object contains the following fields:

Table 89. Format of an equivalent-workload-service-class object

Field name	Type	Description
equiv-workload-name	String	The displayable name of a workload resource group that is part of an equivalent service class that was included in this report.
equiv-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **equiv-workload-name** field.
equiv-service-class-name	String	The displayable name of the equivalent service class (within the workload resource group defined by the **equiv-workload-name**) that was included in this report.
equiv-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **equiv-service-class-name** field.

Each nested hop-entry object contains the following fields:

Table 90. Format of a hop-entry object

Field name	Type	Description
hop-number	Integer	The number of this hop. The hop numbers reflect the relative order of the flow of a work request (transaction) from one application environment to the next. The first hop is 0, the next is hop 1, the next is hop 2, and so on.
hop-name	String	A text name generated for this hop number.
hop-statistics	Object	A hop-report-statistics object that reflects statistics for this hop as a whole (all application environment groups and their virtual servers that are contained in this hop). The format of the returned object is described in Table 91 on page 464.
hop-application-environments	Array of objects	An array of nested hop-application-env objects. The format of the returned object is described in Table 92 on page 464.

Each nested hops-report-statistics object contains the following fields:

Table 91. Format of a hops-report-statistics object

Field name	Type	Description
successful-transactions	Integer	The total number of transactions that completed successfully.
failed-transactions	Integer	The total number of transactions that failed.
stopped-transactions	Integer	Total number of transactions that stopped before completing. These transactions did not fail or complete successfully; a transaction can enter the stopped state if it encounters an error with the application or server that is processing the transaction. For example, if an application detects that its caller or client terminates the request before the transaction instance completes, the application can stop processing for the transaction instance and report it as stopped, rather than failed or successful.
inflight-transactions	Integer	The total number of transactions that had started, but not yet completed by the end of the requested reporting interval. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
queue-time	Integer	The average amount of time (in microseconds) from the time a transaction is received until processing of the transaction begins. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
execution-time	Integer	The average amount of time (in microseconds) that transactions took to execute. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
successful-avg-response-time	Integer	Average response time (in microseconds) of all successful transactions.
inflight-avg-response-time	Integer	Average amount of time (in microseconds) spent toward response time for inflight transactions. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.

Each nested hop-application-env object contains the following fields:

Table 92. Format of a hop-application-env object

Field name	Type	Description
appl-env-name	String	The name of the application environment. An application environment is the environment that includes the software and the server or network infrastructure that supports it. As defined by The Open Group application response measurement (ARM) standard, the name is no more than 128 characters in length.
group-name	String	The name of the application environment group with which this application environment is associated. This value cannot be more than 128 characters long but is zero length when the application environment does not belong to a group.

Table 92. Format of a hop-application-env object (continued)

Field name	Type	Description
`appl-env-hop-statistics`	Object	A hop-statistics object that reflects statistics for this application environment as a whole (all virtual servers that are contained in this application environment). The format of the returned object is described in Table 91 on page 464.
`hop-application-env-virtual-servers`	Array of objects	An array of nested hop-application-env-virtual-server objects. The format of the returned object is described in Table 93.

Each nested hop-application-env-virtual-server object contains the following fields:

Table 93. Format of a hop-application-env-virtual-server object

Field name	Type	Description
`virtual-server-name`	String	The displayable name of the virtual server. This field identifies the virtual server within the application environment.
`virtual-server-hop-statistics`	Object	A hop-report-statistics object that reflects statistics for this specific virtual server within the application environment. The format of the returned object is described in Table 91 on page 464.

Description

The **Generate Service Class Hops Report** operation generates a report that contains the following information for a specific workload resource group and service class over the requested time interval:

- A list of application-level hops associated with this specific workload resource group and service class for which historical reporting information is available. A hop corresponds to a tier in the transactional flow, and each hop can have one or more application environments associated with it.

  A service class can have multiple hops associated with it, and each hop can have one or more associated application environments. These application environments might have multiple virtual servers.
- For each hop associated with the service class:
  - A common set of statistics
  - Statistics for the application environments and virtual servers within each hop.

"Response body contents" on page 462 describes the full list of data that is returned in this report.

For more information about transactional service classes, hops in the transaction flow, and application environments, see the topic "Enhanced monitoring and management through guest platform management providers" in the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. The hops report for the service class can be empty when one of the following circumstances is true for all of the virtual servers associated with the service class:

- A guest platform management provider is not installed or running on the operating system.
- No ARM-instrumented applications are running on the operating system.
- None of the ARM-instrumented applications running on the operating system are hop 0 application environments.

An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements" on page 466.

The request body is validated against the schema described in "Request body contents" on page 461. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Hops Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 462.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-service-class-hops-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group",
 "service-class-report-id": "Batch Service"
}
```

Figure 233. Generate Service Class Hops Report: Request

Generate Service Class Virtual Server Topology Report

Use the **Generate Service Class Virtual Server Topology Report** operation to create a custom on-demand topology report for a specific service class within a workload resource group over a requested time period. The report is based on historical performance management data that was retained over that time period.

HTTP method and URI

POST /api/ensembles/{*ensemble-id*}/performance-management/operations/generate-service-class-virtual-server-topology-report

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble containing the workload resource group for which you want to receive a virtual server topology report.

Request body contents

The request body is a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
report-interval-start-time	Integer	Required	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970.
report-interval-duration	Integer	Required	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0.
workload-report-id	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field. **Note:** This value is the same as one of the **workload-report-id** values that are returned in the Workload Resource Groups Report for this same interval.
service-class-report-id	String	Required	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field. **Note:** This value is the same as the **service-class-report-id** value that is returned in one of the service class entries in the Service Classes Report for this same interval.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
report-interval-start-time	Integer	Standard date/time value indicating the requested starting date and time of the report to be generated. The standard time value is defined as the number of elapsed milliseconds after midnight on 1 January 1970. **Note:** The returned value might differ from the requested time if no report data was available at the requested time, but data was available starting later within the requested interval.
report-interval-duration	Integer	The length, in minutes, of the interval this report covers, starting from the value specified for the **report-interval-start-time** field. This value must be greater than 0. **Note:** The returned value might differ from the requested duration if report data was not available for the entire requested interval, but data was available for a shorter duration within that same requested interval.
workload-name	String	The displayable name of the workload resource group.
workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **workload-name** field.
service-class-name	String	The displayable name of the service class.
service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **service-class-name** field.

Field name	Type	Description
equivalent-workload-service-classes	Array of objects	Because a specific virtual server can belong to more than one workload, the hops report might contain reflect combined values for multiple workload resource groups rather than those for the requested workload resource group and service class. In this case, this field contains an array of nested equivalent-workload-service-class objects, each of which identifies another workload resource group or service class, if any, that have virtual servers also contained within the reported hops. The format of the returned object is described in Table 94. Otherwise, this field contains an empty array.
topo-hop-count	Integer	The total number of hops represented in this topology report.
topo-hops	Array of objects	An array of nested topo-hop objects. The format of the returned object is described in Table 95.

Each nested equivalent-workload-service-class object contains the following fields:

Table 94. Format of an equivalent-workload-service-class object

Field name	Type	Description
equiv-workload-name	String	The displayable name of a workload resource group that is part of an equivalent service class that was included in this report.
equiv-workload-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific workload resource group named in the **equiv-workload-name** field.
equiv-service-class-name	String	The displayable name of the equivalent service class (within the workload resource group defined by the **equiv-workload-name**) that was included in this report.
equiv-service-class-report-id	String	An identifier used by the performance management reporting structure to keep track of reporting data for the specific service class named in the **equiv-service-class-name** field.

Each nested topo-hop object contains the following fields:

Table 95. Format of a topo-hop object

Field name	Type	Description
hop-number	Integer	The number of this hop. The hop numbers reflect the relative order of the flow of a work request (transaction) from one application environment to the next. The first hop is 0, the next is hop 1, the next is hop 2, and so on.
topo-virtual-server-nodes	Array of objects	An array of nested topo-virtual-server-node objects that represent virtual servers that are part of this hop. The format of the returned object is described in Table 96.

Each nested topo-virtual-server-node object contains the following fields:

Table 96. Format of a topo-virtual-server-node object

Field name	Type	Description
node-identifier	String	A topology node identifier that uniquely identifies this specific node within this entire topology report. For example, a virtual server that shows up in multiple hops of the transaction flow has the same virtual server name in all of those nodes but each node identifier is different.
virtual-server-name	String	The displayable name of the virtual server.

Table 96. Format of a topo-virtual-server-node object (continued)

Field name	Type	Description
hypervisor-type	String	The value of the **type** property of the virtualization host. See the virtualization host "Data model" on page 163 for more details about the valid values of this property.
hypervisor-name	String	The displayable name of the hypervisor under which this virtual server was running.
was-active	Boolean	Indicates whether this virtual server was active during this reporting interval. The value is **"true"** if the virtual server was active or **"false"** if it was not active. If the value of this field is false, none of the remaining fields in this table have data, so none of these fields are returned.
physical-cpu-utilization-percent	Number	The physical processor utilization percentage (%) of the virtual server. This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval.
hypervisor-cpu-delay-percent	Number	The hypervisor processing unit delay in fractional form (between 0 and 1 inclusive). This field is returned for the following hypervisors only: z/VM and PowerVM. In the case of z/VM, a value is available for this field only if sampling is turned on for the guest. **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information.
idle-time-percent	Number	The idle time percentage (%) of the virtual server. It is the percentage of total time that the virtual server had no work (that is, no application processes or internal hypervisor specific process states). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information. For z/VM, this data is available only when sampling is enabled and started. For PowerVM and PR/SM, idle time data is not available.
other-time-percent	Number	The percentage (%) of total time that miscellaneous hypervisor specific internal process states had control for this virtual server. In other words, the percentage of time that the virtual server was not idle but also was not in a state of active CPU utilization or hypervisor CPU delay. This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active during this interval and if the hypervisor type supports this information. For z/VM, this data is available only when sampling is enabled and started. For PowerVM and PR/SM, other time data is not available.
os-cpu-using-samples-percent	Number	The percentage of CPU using samples from among the total samples. For example, if there are 10 CPU using samples out of a total of 10 samples, then CPU using samples is 100% (because out of the total samples, all are CPU using samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-cpu-delay-samples-percent	Number	The percentage of CPU delay samples from among the total samples. For example, if there are 10 CPU delay samples and 10 samples that are not CPU delay samples, then CPU delay samples is 50% (because out of the total samples half are CPU delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.

Table 96. Format of a topo-virtual-server-node object (continued)

Field name	Type	Description
os-io-delay-samples-percent	Number	The percentage of I/O delay samples from among the total samples. The percent I/O delay is the percent of samples taken when work was delayed for non-paging DASD I/O. The I/O delay includes IOS queue, subchannel pending, and control unit queue delays. For example, if there are 10 I/O delay samples and 10 samples that are not I/O delay samples, then I/O delay samples is 50% (because out of the total samples half are I/O delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-page-delay-samples-percent	Number	The percentage of page delay samples from among the total samples. The percent page delay is the percent of samples when the address space experienced page faults in cross-memory access, and the page faults were resolved from auxiliary storage. For example, if there are 10 page delay samples and 10 samples that are not page delay samples, then page delay samples is 50% (because out of the total samples half are page delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
appl-env-vs-response-data	Array of objects	An array of nested appl-env-vs-response-entry objects that contain response time data for each of the application environments with which the virtual server is associated. If no response data is available, this field contains an empty array. The format of the returned object is described in Table 97.
appl-env-vs-utilization-data	Array of objects	An array of nested appl-env-vs-utilization-data objects that contain utilization data for each of the application environments with which the virtual server is associated. If no response data is available, this field contains an empty array. The format of the returned object is described in Table 98 on page 471.
child-virtual-server-node-links	Array of objects	An array of nested child-virtual-server-node-link objects that contain a list of links to other virtual server nodes in the topology, if this particular virtual server sent transactions to one or more virtual server nodes in the next hop. These link objects provide the information required to identify those next-level nodes (or "child" virtual servers). If this virtual server node has no child nodes, this field contains an empty array. The format of the returned object is described in Table 99 on page 472.

Each nested appl-env-vs-response-entry object contains the following fields:

Table 97. Format of an appl-env-vs-response-entry object

Field name	Type	Description
appl-env-name	String	The name of the application environment. An application environment is the environment that includes the software and the server or network infrastructure that supports it.
group-name	String	The name of the application environment group with which this application environment is associated. This value cannot be more than 128 characters long but is zero length when the application environment does not belong to a group.
successful-transactions	Integer	The total number of transactions that completed successfully.
failed-transactions	Integer	The total number of transactions that failed.

Table 97. Format of an appl-env-vs-response-entry object (continued)

Field name	Type	Description
`stopped-transactions`	Integer	Total number of transactions that stopped before completing. These transactions did not fail or complete successfully; a transaction can enter the stopped state if it encounters an error with the application or server that is processing the transaction. For example, if an application detects that its caller or client terminates the request before the transaction instance completes, the application can stop processing for the transaction instance and report it as stopped, rather than failed or successful.
`inflight-transactions`	Integer	The total number of transactions that had started, but not yet completed by the end of the requested reporting interval. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
`queue-time`	Integer	The average amount of time (in microseconds) from the time a transaction is received until processing of the transaction begins. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
`execution-time`	Integer	The average amount of time (in microseconds) that transactions took to execute. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.
`successful-avg-response-time`	Integer	Average response time (in microseconds) of all successful transactions.
`inflight-avg-response-time`	Integer	Average amount of time (in microseconds) spent toward response time for inflight transactions. Optional: This data may not always be available. For example, when Linux is involved, it is not usually possible to obtain this statistic. If it is not available, this field will not be returned.

Each nested appl-env-vs-utilization-entry object contains the following fields:

Table 98. Format of an appl-env-vs-utilization-entry object

Field name	Type	Description
`appl-env-name`	String	The name of the application environment. An application environment is the environment that includes the software and the server or network infrastructure that supports it.
`group-name`	String	The name of the application environment group with which this application environment is associated. This value cannot be more than 128 characters long but is zero length when the application environment does not belong to a group.
`cpu-time`	Integer	The processor (CPU) time, in microseconds, that was consumed on this virtual server as part of this application environment.

Table 98. Format of an appl-env-vs-utilization-entry object (continued)

Field name	Type	Description
os-cpu-using-samples-percent	Number	The percentage of CPU using samples from among the total samples. For example, if there are 10 CPU using samples out of a total of 10 samples, then CPU using samples is 100% (because out of the total samples, all are CPU using samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-cpu-delay-samples-percent	Number	The percentage of CPU delay samples from among the total samples. For example, if there are 10 CPU delay samples and 10 samples that are not CPU delay samples, then CPU delay samples is 50% (because out of the total samples half are CPU delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-io-delay-samples-percent	Number	The percentage of I/O delay samples from among the total samples. The percent I/O delay is the percent of samples taken when work was delayed for non-paging DASD I/O. The I/O delay includes IOS queue, subchannel pending, and control unit queue delays. For example, if there are 10 I/O delay samples and 10 samples that are not I/O delay samples, then I/O delay samples is 50% (because out of the total samples half are I/O delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.
os-page-delay-samples-percent	Number	The percentage of page delay samples from among the total samples. The percent page delay is the percent of samples when the address space experienced page faults in cross-memory access, and the page faults were resolved from auxiliary storage. For example, if there are 10 page delay samples and 10 samples that are not page delay samples, then page delay samples is 50% (because out of the total samples half are page delay samples). This percentage is in fractional form (between 0 and 1 inclusive). **Optional:** This field is returned only if the virtual server was active and a guest platform management provider was running on the virtual server during this interval.

Each nested child-virtual-server-node-link object contains the following fields:

Table 99. Format of a child-virtual-server-node-link object

Field name	Type	Description
child-node-identifier	String	The unique identifier assigned within this topology report to this child virtual server node.
child-hop-number	Integer	The number of the hop that contains this child virtual server node.
transaction-count	Integer	The number of transactions that flowed from the parent virtual server node to this specific child virtual server node over this reporting period. If no transaction data is available, this field is not returned.

Description

The **Generate Service Class Virtual Server Topology Report** operation generates a report that contains the following information for a specific service class within a workload resource group over the requested time interval:

- A list of application-level hops associated with this specific workload resource group and service class for which historical reporting information is available. A hop corresponds to a tier in the transactional flow, and each hop can have one or more application environments associated with it.

  A service class can have multiple hops associated with it, and each hop can have one or more associated application environments. These application environments might have multiple virtual servers.
- For each hop associated with the service class:
  - A common set of statistics
  - Statistics for the application environments and virtual servers within each hop.

This report returns information that is similar to the output of the **Generate Service Class Hops Report** operation; the virtual server nodes are organized by hop and the reports contain some similar statistics for each virtual server. This topology report, however, also returns information about the relationship between the virtual servers from one hop to the next. This type of information relationships is not always available, depending on the actual configuration of the application environments.

For an example of a graphical representation of the virtual server topology, use the **Virtual Server Topology Report** that is available through the HMC or see the topic about that report in the *zEnterprise System Ensemble Performance Management Guide*, GC27-2607. The **Virtual Server Topology Report** task graphically depicts the relationships between virtual servers that are running the workload resource group and providing the resources to complete the work. The framed image in that report displays graphical representations of virtual servers. Each is represented by an icon that displays the name and status of the virtual server; by clicking the icon, you can view statistics for that virtual server node. The **Generate Service Class Virtual Server Topology Report** operation generates the same information that you can access through the **Virtual Server Topology Report** and the icons in its display.

"Response body contents" on page 467 describes the full list of data that is returned in the report for this workload resource group.

If reporting data is not available for the requested time interval, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

The request body is validated against the schema described in "Request body contents" on page 467. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Virtual Server Topology Report** and the **View Statistics** tasks.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 467.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-service-class-virtual-server-topology-report
{
 "report-interval-start-time": 1296149252662,
 "report-interval-duration": 60,
 "workload-report-id": "Payroll Workload Resource Group",
 "service-class-report-id": "Batch Service"
}
```

Figure 234. Generate Service Class Virtual Server Topology Report: Request

Generate Load Balancing Report

Use the **Generate Load Balancing Report** operation to list information about network load balancing activity in an ensemble. Unlike other performance management reports, which are based on historical performance management data that was retained over a specific time period, this report contains data that is available when the **Generate Load Balancing Report** API is called.

HTTP method and URI

`POST /api/ensembles/{`*`ensemble-id`*`}/performance-management/operations/generate-load-balancing-report`

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble for which you want to receive a load balancing report.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`load-balancing-groups`	Array of objects	An array of nested load-balancing-group objects. If no load balancing groups have been defined for this ensemble, an empty array is returned for this field.

Each nested load-balancing-group object contains the following fields:

Field name	Type	Description
load-balancer-group-name	String	The name given to this load balancer group. The returned string in this field conforms to Server/Application State Protocol (SASP) standards, which allow a group name to consist of 1 through 255 UTF8 characters.
load-balancer-group-members	Array of objects	An array of nested load-balancer-group-member objects. If a group does not have any members associated with it, an empty array is returned for this field.

Each nested load-balancer-group-member object contains the following fields:

Field name	Type	Description
virtual-server-name	String	The name of a virtual server defined to this load balancing group.
hostname-ipaddr	String	The IP address of the virtual server.
weight	Number	The relative weight recommendation that zManager assigned to this member.
status	String Enum	The status of or environmental conditions for each virtual server in the selected load balancing group. Possible values are: • **"active"** • **"down"** • **"inactive"** • **"failure-recovery"** • **"quiesced"** • **"unknown"**
status-detailed	String Enum	A further explanation for the specific status condition **"down"** or **"inactive"**; for other **status** field values, this field is not returned. Possible values are: • **"matching-ip-address-not-found"** • **"gpmp-downlevel"** • **"lsof-not-available"** • **"no-process-listening"** • **"not-initialized"** • **"vs-not-operating"** • **"load-balance-data-initializing"** • **"unsupported-hypervisor-config"** • **"hypervisor-details-not-available"** • **"data-retrieval-error"**
port-number	Integer	The port number on the virtual server that is being used for load balancing purposes. This field is not returned if the port number is not available.
protocol-type	String Enum	The protocol in use for this virtual server; possible values are: • **"tcp"** for Transmission Control Protocol, or • **"udp"** for User Datagram Protocol This field is not returned if the protocol type is not available.

Description

The **Generate Load Balancing Report** operation generates a report that contains a list of all of the load-balancing groups that zManager is monitoring within a specific ensemble. For each load-balancing group, the report also contains information about each individual virtual server that is a member of the group. “Response body contents” on page 474 describes the full list of data that is returned in this report for each load-balancing group and its members.

The generated load-balancing report contains data that is available when the **Generate Load Balancing Report** API is called. Load-balancing data for an ensemble is refreshed every 30 seconds. For the most efficient use of the **Generate Load Balancing Report** API, space additional API calls at least 30 seconds apart.

If no load-balancing groups are defined for this ensemble, an empty response object is provided and the operation completes successfully. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the ensemble object passed in the request URI.
- Action/task permission to the **Load Balancing Report** task.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 474.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing `ensemble` object, or the API user does not have object access permission to the object.

Example HTTP interaction

```
POST /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/operations/
generate-load-balancing-report
```

Figure 235. Generate Load Balancing Report: Request

Get Performance Management Velocity Level Range Mappings

Use the **Get Performance Management Velocity Level Range Mappings** operation to retrieve the numeric ranges that zManager uses to calculate the actual performance velocity for a service class. Each range of numbers is mapped to one of the valid descriptive values that are used in performance management reports for service classes that have a velocity rather than discretionary performance goal.

HTTP method and URI

`GET /api/ensembles/{`*`ensemble-id`*`}/performance-management/velocity-level-range-mappings`

In this request, the URI variable *{ensemble-id}* is the object ID of the ensemble object.

Response body contents

On successful completion, the response body is a JSON object with the following fields:

Field name	Type	Description
`performance-management-velocity-level-range-mappings`	Array of objects	An array of nested velocity-level-range-mapping-entry objects.

Each nested velocity-level-range-mapping-entry object contains the following fields:

Field name	Type	Description
`low-boundary`	Integer	This value defines the low boundary (inclusive) that defines this specific range of calculated velocity numbers.
`high-boundary`	Integer	This value defines the upper boundary (inclusive) that defines this specific range of calculated velocity numbers.
`velocity-level`	String Enum	The descriptive velocity level that this numeric range represents. Possible values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**.

Description

The **Get Performance Management Velocity Level Range Mappings** operation retrieves the list of numeric ranges that zManager uses to calculate the actual performance for a service class that has a performance goal of the **"velocity"** (rather than **"discretionary"**) type. Each range of numbers is mapped to one of the valid descriptive values that are used in performance management reports; these descriptive values are: **"fastest"**, **"fast"**, **"moderate"**, **"slow"**, and **"slowest"**.

For service classes with a velocity performance goal, zManager:
1. Calculates a numeric value for actual performance
2. Compares the calculated numeric value to a predefined set of ranges, one range for each of the descriptive velocity-level values
3. Substitutes the appropriate descriptive value to present in performance management reports.

This substitution occurs because the actual performance is not significantly different between numbers that are close together; it is more meaningful to define a smaller set of velocity levels that describe the performance.

Through the **Get Performance Management Velocity Level Range Mappings** API and the service class metrics described in "Workload service class data metrics group" on page 659, you can calculate raw velocity numbers and compare and substitute the performance level using the same descriptive velocity levels that zManager uses in its performance management reports.

"Response body contents" describes the full list of data that is returned for the **Get Performance Management Velocity Level Range Mappings** operation. An error response is returned if the targeted ensemble does not exist or if you do not have the requirements listed in "Authorization requirements."

Authorization requirements

This operation has the following authorization requirement:
- object-access permission to the ensemble object passed in the request URI.

HTTP status and reason codes

On successful completion, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 477.

Otherwise, the following HTTP status codes are returned for the indicated errors. The response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/12345678-1234-1234-1234-123456789000/performance-management/
velocity-level-range-mappings
```

Figure 236. Get Performance Management Velocity Level Range Mappings: Request

Inventory service data

Information about the Workload Resource Groups managed by the HMC and the associated Performance Policies can be optionally included in the inventory data provided by the Inventory Service.

Inventory entries for workload and associated policy objects are included in the response to the Inventory Service's Get Inventory operation when the request specifies (explicitly by class, implicitly via a containing category, or by default) that objects of class "workload-resource-group" are to be included. Information for a particular workload (and associated policies) is included only if the API user has object-access permission to that object.

For each workload to be included, the inventory response array includes the following:

- An array entry for the workload object itself. This entry is a JSON object with the same contents as is specified in the Response Body Contents section for "Get Workload Resource Group Properties" on page 375. That is, the data provided is the same as would be provided if a **Get Workload Resource Group Properties** operation were requested targeting this object.
- An array entry for each policy associated with the workload. For each such policy, an entry is included that is a JSON object with the same contents as specified in the Response Body Contents section of "Get Performance Policy Properties" on page 402. As a result, the data provided is the same as would be obtained if a **Get Performance Policy Properties** operation were requested for each policy listed by a **List Performance Policies** operation targeting the workload.

The array entry for a workload object will appear in the results array before entries for associated performance policies.

Sample inventory data

The following fragment is an example of the JSON object that would be included in the **Get Inventory** response to describe a single workload (named "Online Ordering") that defines a custom performance policy ("Prime shift") in addition to the default policy. These objects would appear as a sequence of array entries in the response array:

```
{
   "active-perf-policy": {
      "activation-status": "active",
      "element-id": "cc79445a-95e4-11e0-b6ef-000c29bb873c",
      "element-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c
     /performance-policies/cc79445a-95e4-11e0-b6ef-000c29bb873c",
      "name": "Prime shift"
   },
   "category": "Retail Operations",
   "class": "workload-resource-group",
   "custom-perf-policies": [
      {
         "activation-status": "active",
         "element-id": "cc79445a-95e4-11e0-b6ef-000c29bb873c",
         "element-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c
       /performance-policies/cc79445a-95e4-11e0-b6ef-000c29bb873c",
         "name": "Prime shift"
      }
   ],
   "default-perf-policy": {
      "activation-status": "not-active",
      "element-id": "cc7b38dc-95e4-11e0-b6ef-000c29bb873c",
      "element-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c
     /performance-policies/cc7b38dc-95e4-11e0-b6ef-000c29bb873c",
      "name": "Default"
   },
   "description": "Workload Resource Group for managing the online ordering application.",
   "is-default": false,
   "name": "Online Ordering",
   "object-id": "cc79fa6c-95e4-11e0-b6ef-000c29bb873c",
   "object-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c",
   "parent": "/api/ensembles/890b6df2-93a4-11e0-887c-000c29bb873c",
   "perf-activation-node-count": 0,
   "perf-activation-status": "active"
},
{
   "activation-status": "active",
   "class": "performance-policy",
   "created-by": "ENSADMIN",
   "created-date": 1307987073673,
```

Figure 237. Workload Resource Group: Sample inventory data (Part 1)

```
"custom-service-classes": [
   {
      "business-importance": "highest",
      "classification-rule": {
         "filter": {
            "operation": "string-match",
            "type": "virtual-server-name",
            "value": "OOWS(\\*)"
         },
         "type": "rule"
      },
      "description": "",
      "goal-type": "velocity",
      "name": "Web servers",
      "type": "server",
      "velocity": "fast"
   },
   {
      "business-importance": "highest",
      "classification-rule": {
         "filter": {
            "operation": "string-match",
            "type": "virtual-server-name",
            "value": "OODB(\\*)"
         },
         "type": "rule"
      },
      "description": "",
      "goal-type": "velocity",
      "name": "DB Servers",
      "type": "server",
      "velocity": "fastest"
   }
],
```

Figure 238. Workload Resource Group: Sample inventory data (Part 2)

```
    "default-service-class": {
       "business-importance": "medium",
       "classification-rule": {
          "filter": {
             "operation": "string-match",
             "type": "(\\*)",
             "value": "(\\*)"
          },
          "type": "rule"
       },
       "description": "The default workload performance policy service class.",
       "goal-type": "velocity",
       "name": "Default",
       "type": "server",
       "velocity": "moderate"
    },
    "description": "Performance policy for prime shift operatoin",
    "element-id": "cc79445a-95e4-11e0-b6ef-000c29bb873c",
    "element-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-
    000c29bb873c/performance-policies/cc79445a-95e4-11e0-b6ef-000c29bb873c",
    "importance": "highest",
    "is-default": false,
    "last-activated-by": "PEDEBUG",
    "last-activation-completed-date": 1307991376227,
    "last-activation-requested-date": 1307991376226,
    "last-modified-by": "ENSADMIN",
    "last-modified-date": 1307987073673,
    "name": "Prime shift",
    "parent": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c",
    "revision": 1
 },
 {
    "activation-status": "not-active",
    "class": "performance-policy",
    "created-by": "",
    "created-date": -1,
    "custom-service-classes": [],
```

Figure 239. Workload Resource Group: Sample inventory data (Part 3)

```
    "default-service-class": {
       "business-importance": "medium",
       "classification-rule": {
          "filter": {
             "operation": "string-match",
             "type": "(\\*)",
             "value": "(\\*)"
          },
          "type": "rule"
       },
       "description": "The default workload performance policy service class.",
       "goal-type": "velocity",
       "name": "Default",
       "type": "server",
       "velocity": "moderate"
    },
    "description": "The default workload performance policy",
    "element-id": "cc7b38dc-95e4-11e0-b6ef-000c29bb873c",
    "element-uri": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c
    /performance-policies/cc7b38dc-95e4-11e0-b6ef-000c29bb873c",
    "importance": "medium",
    "is-default": true,
    "last-activated-by": "PEDEBUG",
    "last-activation-completed-date": 1307991361463,
    "last-activation-requested-date": 1307991361463,
    "last-modified-by": "",
    "last-modified-date": -1,
    "name": "Default",
    "parent": "/api/workload-resource-groups/cc79fa6c-95e4-11e0-b6ef-000c29bb873c",
    "revision": 1
}
```

Figure 240. Workload Resource Group: Sample inventory data (Part 4)

Chapter 14. Core System z resources

These APIs provide access to and control of the following HMC/SE objects:

- Console
- Group
- CPC
- Logical Partition

In addition, these APIs provide access to the following CPC-related data items:

- Reset Activation Profiles
- Image Activation Profiles
- Load Activation Profiles
- Group Profiles
- Capacity Records

Operations Summary

Following are the operations summaries for the Console, Custom Groups, CPC, Logical Partitions, Activation Profile, and Capacity Record objects.

Console operations summary

The following table provides an overview of the operations provided for Console objects.

Table 100. Core System z resources - Console: operations summary

Operation name	HTTP method and URI path
"Get Console Properties" on page 492	`GET /api/console`
"Restart Console" on page 496	`POST /api/console/operations/restart`
"Make Console Primary" on page 498	`POST /api/console/operations/make-primary`
"Shutdown Console" on page 499	`POST /api/console/operations/shutdown`

Custom groups operations summary

The following table provides an overview of the operations provided for Group objects.

Table 101. Core System z resources - Custom groups: operations summary

Operation name	HTTP method and URI path
"List Custom Groups" on page 502	`GET /api/groups`
"Get Custom Group Properties" on page 504	`GET /api/groups/{group-id}`
"Create Custom Group" on page 505	`POST /api/groups`

Table 101. Core System z resources - Custom groups: operations summary (continued)

Operation name	HTTP method and URI path
"Delete Custom Group" on page 507	`DELETE /api/groups/{group-id}`
"List Custom Group Members" on page 512	`GET /api/groups/{group-id}/members`
"Add Member to Custom Group" on page 508	`POST /api/groups/{group-id}/operations/add-member`
"Remove Member from Custom Group" on page 510	`POST /api/groups/{group-id}/operations/remove-member`

Table 102. Core System z resources - Custom groups: URI variables

URI variable	Description
{group-id}	Object ID of a Group object

CPC operations summary

The following tables provide an overview of the operations provided for CPC objects.

Table 103. Core System z resources - CPC: operations summary

Operation name	HTTP method and URI path
"List CPC Objects" on page 523	`GET /api/cpcs`
"List Ensemble CPC Objects" on page 525	`GET /api/ensembles/{ensemble-id}/cpcs`
"Get CPC Properties" on page 527	`GET /api/cpcs/{cpc-id}`
"Update CPC Properties" on page 533	`POST /api/cpcs/{cpc-id}`
"Activate CPC" on page 535	`POST /api/cpcs/{cpc-id}/operations/activate`
"Deactivate CPC" on page 537	`POST /api/cpcs/{cpc-id}/operations/deactivate`
"Import Profiles" on page 539	`POST /api/cpcs/{cpc-id}/operations/import-profiles`
"Export Profiles" on page 540	`POST /api/cpcs/{cpc-id}/operations/export-profiles`
"Add Temporary Capacity" on page 541	`POST /api/cpcs/{cpc-id}/operations/add-temporary-capacity`
"Remove Temporary Capacity" on page 543	`POST /api/cpcs/{cpc-id}/operations/remove-temporary-capacity`
"Swap Current Time Server" on page 545	`POST /api/cpcs/{cpc-id}/operations/swap-cts`
"Set STP Configuration" on page 546	`POST /api/cpcs/{cpc-id}/operations/set-stp-config`
"Change STP-only Coordinated Timing Network" on page 547	`POST /api/cpcs/{cpc-id}/operations/change-stponly-ctn`

Table 103. Core System z resources - CPC: operations summary (continued)

Operation name	HTTP method and URI path
"Join STP-only Coordinated Timing Network" on page 549	`POST /api/cpcs/{cpc-id}/operations/join-stponly-ctn`
"Leave STP-only Coordinated Timing Network" on page 550	`POST /api/cpcs/{cpc-id}/operations/leave-stponly-ctn`

Table 104. Core System z resources - CPC: URI variables

URI variable	Description
{cpc-id}	Object ID of a CPC object
{ensemble-id}	Object ID of an ensemble object

Logical partitions operation summary

The following tables provide an overview of the operations provided for Logical Partition objects.

Table 105. Core System z resources - Logical partitions: operations summary

Operation name	HTTP method and URI path
"List Logical Partitions of CPC" on page 564	`GET /api/cpcs/{cpc-id}/logical-partitions`
"Get Logical Partition Properties" on page 566	`GET /api/logical-partitions/{logical-partition-id}`
"Update Logical Partition Properties" on page 569	`POST /api/logical-partitions/{logical-partition-id}`
"Activate Logical Partition" on page 570	`POST /api/logical-partitions/{logical-partition-id}/operations/activate`
"Deactivate Logical Partition" on page 572	`POST /api/logical-partitions/{logical-partition-id}/operations/deactivate`
"Reset Normal" on page 574	`POST /api/logical-partitions/{logical-partition-id}/operations/reset-normal`
"Reset Clear" on page 576	`POST /api/logical-partitions/{logical-partition-id}/operations/reset-clear`
"Load Logical Partition" on page 578	`POST /api/logical-partitions/{logical-partition-id}/operations/load`
"PSW Restart" on page 580	`POST /api/logical-partitions/{logical-partition-id}/operations/psw-restart`
"Start Logical Partition" on page 581	`POST /api/logical-partitions/{logical-partition-id}/operations/start`
"Stop Logical Partition" on page 583	`POST /api/logical-partitions/{logical-partition-id}/operations/stop`
"SCSI Load" on page 584	`POST /api/logical-partitions/{logical-partition-id}/operations/scsi-load`
"SCSI Dump" on page 586	`POST /api/logical-partitions/{logical-partition-id}/operations/scsi-dump`

Table 106. Core System z resources - Logical partitions: URI variables

URI variable	Description
{cpc-id}	Object ID of a CPC object
{logical-partition-id}	Object ID of a Logical Partition object

Activation profile operations summary

The following tables provide an overview of the operations provided for the various types of Activation Profile objects.

Table 107. Core System z resources - Reset activation profile: operations summary

Operation name	HTTP method and URI path
"List Reset Activation Profiles" on page 589	`GET /api/cpcs/{cpc-id}/reset-activation-profiles`
"Get Reset Activation Profile Properties" on page 591	`GET /api/cpcs/{cpc-id}/reset-activation-profiles/{reset-activation-profile-name}`
"Update Reset Activation Profile Properties" on page 593	`POST /api/cpcs/{cpc-id}/reset-activation-profiles/{reset-activation-profile-name}`

Table 108. Core System z resources - Image activation profile: operations summary

Operation name	HTTP method and URI path
"List Image Activation Profiles" on page 609	`GET /api/cpcs/{cpc-id}/image-activation-profiles`
"Get Image Activation Profile Properties" on page 611	`GET /api/cpcs/{cpc-id}/image-activation-profiles/{image-activation-profile-name}`
"Update Image Activation Profile Properties" on page 614	`POST /api/cpcs/{cpc-id}/image-activation-profiles/{image-activation-profile-name}`

Table 109. Core System z resources - Load activation profile: operations summary

Operation name	HTTP method and URI path
"List Load Activation Profiles" on page 618	`GET /api/cpcs/{cpc-id}/load-activation-profiles`
"Get Load Activation Profile Properties" on page 620	`GET /api/cpcs/{cpc-id}/load-activation-profiles/{load-activation-profile-name}`
"Update Load Activation Profile Properties" on page 621	`POST /api/cpcs/{cpc-id}/load-activation-profiles/{load-activation-profile-name}`

Table 110. Core System z resources - Group profile: operations summary

Operation name	HTTP method and URI path
"List Group Profiles" on page 623	`GET /api/cpcs/{cpc-id}/group-profiles`
"Get Group Profile Properties" on page 625	`GET /api/cpcs/{cpc-id}/group-profiles/{group-profile-name}`
"Update Group Profile Properties" on page 627	`POST /api/cpcs/{cpc-id}/group-profiles/{group-profile-name}`

Table 111. Core System z resources - Activation profile: URI variables

URI variable	Description
{cpc-id}	Object ID of a CPC object

Table 111. Core System z resources - Activation profile: URI variables (continued)

URI variable	Description
{group-profile-name}	Group profile name
{image-activation-profile-name}	Image activation profile name
{load-activation-profile-name}	Load activation profile name
{reset-activation-profile-name}	Reset activation profile name

Capacity record operations summary

The following tables provide an overview of the operations provided for Logical Partition objects.

Table 112. Core System z resources - Capacity record: operations summary

Operation name	HTTP method and URI path
"List Capacity Records" on page 630	`GET /api/cpcs/{cpc-id}/capacity-records`
"Get Capacity Record Properties" on page 631	`GET /api/cpcs/{cpc-id}/capacity-records/{capacity-record-id}`

Table 113. Core System z resources - Capacity record: URI variables

URI variable	Description
{cpc-id}	Object ID of a CPC object
{capacity-record-id}	Capacity record identifier

Shared nested objects

Some of the Core API objects share common nested objects and are documented here for ease of reference.

Table 114. ec-mcl-description object

Field name	Type	Description
actions	Array of action objects	An optional array of pending action objects. This field is only provided when the HMC is communicating with the CPC's SE.
ec	Array of ec objects	An optional array of EC objects. This field is only provided when the HMC is communicating with the CPC's SE.

Table 115. action object

Field name	Type	Description
type	String Enum	One of: • **"channel-config"** - channels pending a config on/off • **"coupling-facility-reactivation"** - at least one coupling facility pending reactivation • **"power-on-reset-tracking"** - there is a need for a power-on-reset • **"zhybrid-blades-activation"** - (zHybrid accelerator blades are pending an activation.
activation	String Enum	One of: • **"current"** - the action is for the current activation • **"next"** - the action is for the next install and activation.

Table 115. action object (continued)

Field name	Type	Description
pending	Boolean	Is the action pending (true) or not pending (false)

Table 116. ec object

Field name	Type	Description
number	String (1-6)	Engineering Change stream identifier.
part-number	String (1-8)	Engineering Change stream part number.
type	String (1-32)	Engineering Change stream name.
description	String (1-65)	Engineering Change stream descriptive text.
mcl	Aray of mcl objects	The list of MicroCode Levels for this Engineering Change.

Table 117. mcl object

Field name	Type	Description
type	String Enum	One of: • **"retrieved"** - a retrieved or staged level • **"activated"** - an activated or applied level • **"accepted"** - a committed level • **"installable-concurrent"** - a non-disruptive apply-able level • **"removable-concurrent"** - a non-disruptive reject-able level.
level	String (1-3)	Microcode level.
last-update	Timestamp	Time stamp of the last update, in the number of milliseconds since midnight January 1, 1970 UTC. A null object is returned if no updates have occurred.

Table 118. stp-config object

Field name	Type	Description
stp-id	String (1-8)	If in STP-only or Mixed CTN, the STP identifier. Otherwise, an empty string. Valid characters are 0-9, a-z, A-Z, underscore(_) and dash(-).
etr-id	Integer (0-31)	ETR Identifier, if in ETR mode. If not in ETR mode, a null object is returned.
preferred-time-server	stp-node-object	Describes the Preferred Timer Server. This property is optional, only returned on a Get request when the information is set.
backup-time-server	stp-node-object	Describes the Backup Timer Server. This property is optional, only returned on a Get request when the information is set.
arbiter	stp-node-object	Describes the arbiter of the CTN. This property is optional, only returned on a Get request when the information is set.
current-time-server	String Enum	Describes the CPC's role in the CTN. One of: • **"preferred"** - CPC is the Preferred Time Server • **"backup"** - CPC is the Backup Time Server.

This object is used to identify a CPC to the STP services. When used as input on the Set STP Configuration operation, if the **object-uri** field is not provided, all other fields are required. If the **object-uri** field is provided, all other fields are optional. If all fields are provided, the **object-uri** field is ignored.

Table 119. stp-node object

Field name	Type	Description
object-uri	String URI	If the CPC is known to the HMC, contains the CPC's object-uri. Otherwise, contains a null object
type	String (0-6)	The CPC machine type, right justified and left padded with zeros or a empty string
model	String (0-3)	The CPC machine model or a empty string
manuf	String (0-3)	The CPC manufacturer or a empty string
po-manuf	String (0-2)	The CPC plant of manufacturer or a empty string
seq-num	String (0-12)	The CPC sequence number or a empty string

Table 120. psw-description object

Field name	Type	Description
psw	String	Program Status Word (PSW) information for a single processor.
cpid	String (2)	The hexadecimal processor identifier, right justified and left padded with zeros.

Table 121. zaware-network object

Field name	Type	Description
chpid	String (2)	The required network adapter channel path identifier, in hexadecimal characters 0-9,a-f,A-F. The format is two hexadecimal digits (00-FF).
ipaddr-type	String Enum	Indicates how this network adapter's IP address is obtained. One of the following values: • **"dhcp"** - obtains an IP address via DHCP • **"link-local"** - obtains a link-local IP address • **"static"** - uses the specified IP address information.
vlan-id	Integer (0-65535)	If this network adapter is attached to a Virtual LAN, this field contains the VLAN identifier. Otherwise, a null object indicating that this network adapter is not attached to a Virtual LAN.
static-ip-info	network-ip-info object	When **ipaddr-type** is **"static"**, contains the static IP information. Otherwise, contains a null object indicating that a static IP address is not used.

Table 122. ip-info object

Field name	Type	Description
type	String Enum	The type of IP address being provided. One of the following values: • **"ipv4"** - an IPv4 address is provided • **"ipv6"** - an IPv6 address is provided.
ip-address	String/ IPV4 address or String/ IPV6 address	The IP address to be used. The format of the string (IPv4 or IPv6) must be as indicated by the **type** field.

Table 123. network-ip-info object

Field name	Type	Description
type	String Enum	The type of IP address being provided. One of the following values: • **"ipv4"** - an IPv4 address is provided • **"ipv6"** - an IPv6 address is provided.
ip-address	String/ IPV4 address or String/ IPV6 address	The IP address to be used. The format of the string (IPv4 or IPv6) must be as indicated by the **type** field.
prefix	Integer	The number of leading bits of ip-address that represent the network prefix. • When **type** is **"ipv4"** - valid values are 0-32 • When **type** is **"ipv6"** - valid values are 0-128.

Console object

The Console object represents the single zEnterprise Hardware Management Console (HMC) application. The Console object offers a heterogeneous set of services and capabilities, from basic HMC control operations to general HMC information.

Object access to the single Console object representing the local HMC is automatic for all authenticated users. A Console may (but need not) participate in a { Primary, Alternate } pairing with another Console. The Console object helps facilitate the management of a HMC's role in such a pairing.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, but does not provide the operational-status-related properties defined in that schema because it does not maintain the concept of an operational status.

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

The following class-specific specializations apply to the other Base Managed Object properties:

Table 124. Console object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/URI	The canonical URI path of the Console object, of the form `/api/console`
parent	—	String/URI	A Console object has no parent, so this property is always a null object.
class	—	String	The class of a Console object is **"console"**.
name	(ro)	String	The installation assigned name
description	(ro)	String	This property is not supported, and returned as null.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 125. Console object: class specific additional properties

Name	Type	Description
version	String (1-8)	The version number for the Console object.
paired-role	String Enum	An indication (**"primary"** or **"alternate"**) of how the Console object functions in a pairing, or null if this Console object is not paired.
ec-mcl-description	ec-mcl-description object	A nested object that describes the EC (Engineering Change) and MCL (Microcode Level) for the Console. Refer to the description of the ec-mcl-description object for details.
is-auto-switch-enabled	Boolean	Automatic switching between primary and alternate Hardware Management Consoles is enabled (true), or is not enabled (false).
network-info	network-info object	A nested object describing the network information for this Hardware Management Console and for any other Hardware Management console with which this Hardware Management Console may be paired.
machine-info	machine-info object	A nested object describing the machine's BIOS characteristics.
ip-swapping-available	Boolean	Whether the current primary HMC IP addresses will be moved when making the alternate HMC the new primary (true) or not (false).

Table 126. network-info object properties

Name	Type	Description
this-hmc	Array of detailed-network-info objects	The collection of network information for the local Hardware Management Console. The number of objects returned is a function of the machine model and type on which the Hardware Management Console is executing. This information is available in the machine-info property.
paired-hmc	paired-ip-info object	Describes the IP information for the paired Hardware Management Console known to the local Hardware Management Console

Table 127. detailed-network-info properties

Name	Type	Description
hmc-name	String (1-16)	The Hardware Management Console name
interface-name	String	The network interface name
domain-name	String (1-255)	The domain name configured for this network interface
is-private	Boolean	Whether the interface is private (true) or public (false).
mac	String (1-12)	The MAC address of this network interface.
ipv4-address	Array of ipv4-info objects	A collection of nested objects which describe the IPv4 addresses for this network interface.
ipv6-address	Array of ipv6-info objects	A collection of nested objects which describe the IPv6 addresses for this network interface.

Table 128. paired-ip-info properties

Name	Type	Description
hmc-name	String (1-16)	The Hardware Management Console name

Table 128. paired-ip-info properties (continued)

Name	Type	Description
ipv4-address	Array of String IPV4 address	The collection of IPv4 addresses
ipv6-address	Array of String IPV6 address	The collection of IPv6 addresses

Table 129. ipv4-info properties

Name	Qualifier	Type	Description
subnet-mask	(pc)	String (1-15)	The IP mask value
ip-address	(pc)	String IPV4 address	The IPv4 address

Table 130. ipv6-info properties

Name	Qualifier	Type	Description
prefix-length	(pc)	Integer	The number of leading bits of the IPv6 address that represent the network prefix.
ip-address	(pc)	String IPV6 address	The IPv6 address

Table 131. machine-info properties

Name	Type	Description
machine-type	String (1-4)	The type of machine on which the Hardware Management Console is executing.
machine-model	String (1-3)	The model of machine on which the Hardware Management Console is executing.
machine-serial	String (1-10)	The serial number of machine on which the Hardware Management Console is executing.

Get Console Properties

The **Get Console Properties** operation retrieves the properties of the Console object.

HTTP method and URI

```
GET /api/console
```

Response body contents

On successful completion, the response body contains an object that provides the current values of the properties for the Console object as defined in "Data model" on page 490. Field names and data types in the object are the same as the property names and data types defined in the data model.

Description

This operation returns the current properties for the Console object.

This operation may be targeted to an alternate Hardware Management Console.

On successful execution, HTTP status code 200 (OK) is returned and all of the current properties as defined by the Data Model for the Console object are provided in the response body.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described "Response body contents" on page 492.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/console HTTP/1.1
x-api-session: 6djngfmjus22whw3b5oj670c09rb2izzaafr4t3i1iw60ujmkd
```

Figure 241. Get Console Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:04:31 GMT
content-type: application/json;charset=UTF-8
content-length: 4250
{
   "class": "console",
   "description": "R32 Primary/Alternate HMC",
   "ec-mcl-description": {
      "ec": [
         {
            "description": "Hardware Management Console Framework",
            "mcl": [
               {
                  "last-update": "2011-11-09T15:06:34Z",
                  "level": "205",
                  "type": "retrieved"
               },
               {
                  "last-update": "2011-11-09T15:09:07Z",
                  "level": "205",
                  "type": "activated"
               },
               {
                  "last-update": "2011-11-09T15:05:51Z",
                  "level": "167",
                  "type": "accepted"
               },
               {
                  "level": "205",
                  "type": "installable-concurrent"
               },
               {
                  "level": "168",
                  "type": "removable-concurrent"
               }
            ],
            "number": "N48180",
            "part-number": "45D8928",
            "type": "SYSTEM"
         },
```

Figure 242. Get Console Properties: Response (Part 1)

```
        {
           "description": "Embedded Operating System",
           "mcl": [
              {
                 "level": "000",
                 "type": "retrieved"
              },
              {
                 "level": "000",
                 "type": "activated"
              },
              {
                 "level": "000",
                 "type": "accepted"
              },
              {
                 "level": "000",
                 "type": "installable-concurrent"
              },
              {
                 "level": "000",
                 "type": "removable-concurrent"
              }
           ],
           "number": "N48198",
           "part-number": "45D8929",
           "type": "OS"
        }
     ]
},
"ip-swapping-available": true,
"is-auto-switch-enabled": true,
"is-locked": false,
"machine-info": {
   "machine-model": "PAA",
   "machine-serial": "KQZBLKD",
   "machine-type": "7327"
},
"name": "HMCR32PRI",
"network-info": {
   "paired-hmc": {
      "hmc-name": "HMCR32ALT",
      "ipv4-address": [
         "9.60.15.47",
         "9.60.14.47"
      ],
      "ipv6-address": [
         "fdd8:673b:d89b:1:221:5eff:fe69:e3f5",
         "2002:93c:ffb:1:221:5eff:fe69:e3f5",
         "fe80:0:0:0:221:5eff:fe69:e3f5",
         "fe80:0:0:0:210:18ff:fe4c:8026"
      ]
   },
```

Figure 243. Get Console Properties: Response (Part 2)

```
    "this-hmc": [
        {
           "domain-name": "endicott.ibm.com",
           "hmc-name": "HMCR32PRI",
           "interface-name": "eth0",
           "ipv4-address": [
              {
                 "ip-address": "9.60.15.48",
                 "subnet-mask": "255.255.255.0"
              }
           ],
           "ipv6-address": [
              {
                 "ip-address": "fdd8:673b:d89b:1:221:5eff:fe69:dea0",
                 "prefix-length": 64
              },
              {
                 "ip-address": "2002:93c:ffb:1:221:5eff:fe69:dea0",
                 "prefix-length": 64
              },
              {
                 "ip-address": "fe80:0:0:0:221:5eff:fe69:dea0",
                 "prefix-length": 64
              }
           ],
           "is-private": false,
           "mac": "00215E69DEA0"
        },
        {
           "domain-name": "endicott.ibm.com",
           "hmc-name": "HMCR32PRI",
           "interface-name": "eth1",
           "ipv4-address": [
              {
                 "ip-address": "9.60.14.48",
                 "subnet-mask": "255.255.255.0"
              }
           ],
           "ipv6-address": [
              {
                 "ip-address": "fe80:0:0:0:210:18ff:fe4c:8334",
                 "prefix-length": 64
              }
           ],
           "is-private": false,
           "mac": "0010184C8334"
        }
     ]
  },
  "object-id": "1e8b3137-85b0-3a06-9269-0b25fc170d44",
  "object-uri": "/api/console",
  "paired-role": "primary",
  "parent": null,
  "version": "2.11.1"
}
```

Figure 244. Get Console Properties: Response (Part 3)

Restart Console

The **Restart Console** operation restarts the Hardware Management Console.

HTTP method and URI

```
POST /api/console/operations/restart
```

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	Whether the restart operation is processed when users are connected (true) or not (false). The default is false.

Description

The Hardware Management Console is restarted. This operation may be targeted to an alternate Hardware Management Console.

By default, the restart does not occur if one or more users are currently connected to the Hardware Management Console. This can be overridden by use of the **force** field in the request body.

On success, HTTP status code 202 (Accepted) is returned.

Authorization

To use **Restart Console**, you must have the following:

- Action/Task permission to the **Shutdown/Restart** task
- Remote Restart must be enabled on the Hardware Management Console.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See"Common request validation reason codes" on page 18 for a list of the possible reason codes.
	267	The operation is rejected, due to the presence of HMC users. Either wait until all HMC users have logged off or retry the request with the **force** field set to **"true"**.
403 (Forbidden)	269	This operation is currently blocked. The error message will contain information on the blocking application.
	270	The remote restart operation is not enabled on the HMC.
	1	The remote restart operation is not enabled on the HMC.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Make Console Primary

Make Console Primary initiates a primary/alternate role switch when directed at a Hardware Management Console that is currently operating in the alternate role. It is only supported for a Hardware Management Console participating in a {primary, alternate} pairing.

HTTP method and URI

```
POST /api/console/operations/make-primary
```

Description

This operation initiates a primary/alternate role switch when directed at a Hardware Management Console that is currently operating in the alternate role. As this command will cause both consoles to be rebooted, any active sessions (GUI based or API based) will be terminated.

If this operation is directed at a Hardware Management Console which does not participate in a {primary, alternate} pairing, it returns with HTTP status code 400 (Bad Request).

If this operation is directed at a Hardware Management Console whose **pair-role** is **"alternate"**, it initiates the role-switch process, and returns with HTTP status code 204 (No Content).

If this operation is directed at a Hardware Management Console whose **pair-role** is already **"primary"**, it has no effect, and returns with HTTP status code 204 (No Content).

Authorization

To use **Make Console Primary**, you must have the following:
- Action/task permission to the **Manage Alternate HMC** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	265	The operation was directed at a Hardware Management Console which is not participating in a {primary, alternate} pairing, which is not supported.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

When configured as recommended by IBM, the process of recovering from the failure of the primary HMC by takeover by the alternate HMC includes movement of the IP address of the former primary HMC to the new primary HMC. When this occurs, explicit redirection of API requests to the newly

designated primary HMC is not needed. However, the IP address swapping may not be possible in certain network configurations. The address of the alternate HMC is provided to allow applications to explicitly redirect requests to the other HMC of the pair in these cases.

Shutdown Console

Shutdown Console powers off the Hardware Management Console.

HTTP method and URI

```
POST /api/console/operations/shutdown
```

Request body contents

A request body must be specified. It has the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	Whether the shutdown operation is processed when users are connected (true) or not (false). The default is false.

Description

The Hardware Management Console is powered off.

This operation may be targeted to an alternate Hardware Management Console.

By default, the shutdown does not occur if one or more users are currently connected to the Hardware Management Console. This can be overridden by use of the force field in the request body.

The action to shutdown the Hardware Management Console occurs asynchronously. If the request is accepted, HTTP status code 202 (Accepted) is returned to indicate that the request has been initiated. However, because this action results in the targeted Hardware Management Console becoming inactive and powered off at completion, it is not possible to track the completion of this request. Thus no response body containing an asynchronous job URI is provided, nor is a job completion notification generated upon completion.

Authorization

To use **Shutdown Console**, you must have the following:

- Action/task permission to the **Shutdown/Restart** task.
- Remote Shutdown must be enabled on the Hardware Management Console.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned but no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	267	The operation is rejected, due to the presence of HMC users. Either wait until all HMC users have logged off or retry the request with the force field set to true.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
	270	The remote restart operation is not enabled on the HMC.
	304	This operation is currently blocked. The error message will contain information on the blocking application.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
POST /api/console/operations/shutdown HTTP/1.1/
x-api-session: 5dul8zvlwa5s83eobcukaf1vug3s3kgidkyk9e5c5acsekabsl
content-type: application/json
content-length: 16
{
   "force": false
}
```

Figure 245. Shutdown Console: Request

```
202 Accepted
server: zSeries management console API web server / 2.0
cache-control: no-cache
date: Fri, 01 Mar 2013 19:38:25 GMT

<No response body>
```

Figure 246. Shutdown Console: Response

Group Object

The Hardware Management Console provides a fixed set of system-defined groups to which managed objects of certain types automatically belong, as members. For example, defined CPCs are automatically members of the CPC group.

These system-defined groups are distinct from user-defined (“custom”) groups. The latter are explicitly created by users for their own purposes: for example, it may be convenient for management purposes to take some proper subset of the members of the system-defined CPC group as a user-defined group of CPCs. User-defined groups may be homogeneous (all members of the same managed object type, as in this previous example), but need not be.

This API pertains only to user-defined groups. This is because:

- By their nature, the members of the system-defined groups are already obtainable via list operations of the appropriate API. For example, all the CPCs managed by a Hardware Management Console can be obtained via a List CPCs operation. Therefore, list operations for system-defined groups are unnecessary.
- By their nature, the existence of a system-defined group and its content (members) is implicit. Therefore, create/delete operations for system-defined groups are both unnecessary and inappropriate.

A Group object represents one or more managed objects which are called group members. Each member is of some object type: CPC, Logical Partition, etc. Note that groups may be heterogeneous (with member objects of differing types), and may even have other groups as members.

Users may define groups in one of two ways:

1. by use of a pattern-match expression to implicitly define membership (pattern-matching group)
2. by explicitly choosing members.

This API can be used to view/manage custom groups, and membership within these groups. The latter is subject to restrictions, based on which of the two fundamentally different means of definition the user employed:

- If pattern-matching was specified, then group membership is "implicit". In this case, operations to add/remove a member are unnecessary (simply create/delete the managed object, itself, using the appropriate API operation). Accordingly, member-management operations are not supported for groups using pattern-matching.
- If pattern-matching was not specified, then group membership is "explicit", and in this case operations to add/remove group members are both useful and appropriate. Accordingly, for custom groups not based on pattern-matching, member-management operations are supported. Note that such operations do not affect the member object itself, only its group membership status.
- When groups are defined using pattern-matching, the types of managed objects to which pattern-matching is applied must be explicitly specified. Regardless of the POSIX regular expression specified as the match pattern, managed objects whose names match the pattern but who are not of the specified object type(s) are not considered to be members of the group.
- Groups are not intrinsically ordered in any way, nor are members within a given group. List-oriented operations therefore do not return ordered results.

Data model

This object includes the properties defined in the "Base managed object properties schema" on page 33, but does not provide the operational-status-related properties defined in that schema because it does not maintain the concept of an operational status.

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32

The following class-specific specializations apply to the other Base Managed Object properties:

Table 132. Group object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path of the Group object, of the form `/api/groups/{group-id}` where *{group-id}* is the value of the **object-id** property of the Group object.
parent	—	String	This property is always a null object.
class	—	String	The class of a Group object is **"group"**.
name	(ro)	String	The group name specified by the user when the group was created

Table 132. Group object: base managed object properties specializations (continued)

Name	Qualifier	Type	Description of specialization
description	(ro)	String	The description specified by the user when the group was created, or if none was provided, the IBM provided caption text.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties:

Table 133. Group object: class specific additional properties

Name	Type	Description
match-info	match-info object	A nested object which pertains to pattern-matching groups only, as described in the next table. An empty value is returned for groups which do not use pattern-matching.

The match-info object contains the following fields:

Table 134. match-info object properties

Name	Type	Description
pattern	String	A regular expression used to define membership for pattern-matching groups. This field has no length limitations.
types	Array of String Enum	Specifies the type(s) of objects that are eligible for membership in pattern-matching groups. One or more of the following: • **"custom-groups"** - zManager API objects of class “group” • **"defined-cpc"** - zManager API objects of class “cpc” • **"director-timer-console"** - ESCON® director and/or Sysplex timer consoles • **"ibm-fiber-saver"** - IBM 2029 fiber optic data transports • **"logical-partition"** - zManager API objects of class “logical-partition” • **"zvm-virtual-machines"** - zManager API objects of class “virtual-server”, type of **"zvm"** • **"blade-center"** - zManager API objects of class “bladecenter” • **"data-power-xi50z-blades"** - zManager API objects of class “blade”, type of **"dpxi50z"** • **"ibm-smart-analytics-optimizer-blades"** - zManager API objects of class “blade”, type of **"isaopt"** • **"power-blade"** - zManager API objects of class “blade”, type of **"power"** • **"system-x-blade"** - zManager API objects of class “blade”, type of **"x-hyp"** • **"power-vm-virtual-server"** - zManager API objects of class “virtual-server”, type of **"power-vm"** • **"system-x-virtual-server"** - zManager API objects of class “virtual-server”, type of **"x-hyp"** • **"workload"** - zManager API objects of class “workload”.

List Custom Groups

The **List Custom Groups** operation lists the custom groups which are visible to the API user.

HTTP method and URI

```
GET /api/groups
```

Query Parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`groups`	Array of group-info objects	Array of nested objects which identify custom groups that are visible to the API user.

Each nested group-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/URI	The value of the Group object's **object-uri** property.
`name`	String	The value of the Group object's **name** property.

Description

This operation lists the custom group objects which are visible to the API user. Only groups to which the caller has authorization will be returned.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section. If no custom groups exist, or if no custom groups are visible to the API user, HTTP status code 200 (OK) is returned, along with an empty response body.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the group object.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Example HTTP interaction

```
GET /api/groups HTTP/1.1
x-api-session: 4ipkcgbjpy5kocelt652l3dvb85gi81iqy5bz8yrpt6vtrt8ks
```

Figure 247. List Custom Groups: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:02:42 GMT
content-type: application/json;charset=UTF-8
content-length: 283
{
   "groups": [
      {
         "name": "SS-Web-Servers",
         "object-uri": "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341"
      },
      {
         "name": "Test Group",
         "object-uri": "/api/groups/febde5ab-a4a6-35bf-9e01-83aae59d7e52"
      }
   ]
}
```

Figure 248. List Custom Groups: Response

Get Custom Group Properties

The **Get Custom Group Properties** operation retrieves the properties of a single custom group object that is designated by the *{group-id}*.

HTTP method and URI

GET /api/groups/{*group-id*}

In this request, the URI variable *{group-id}* is the object ID of the group.

Response body contents

On success, HTTP status code 200 (OK) is returned and the response body contains an object that provides the current values of the properties for the Group object as defined in “Data model” on page 501. Field names and data types in the object are the same as the property names and data types defined in the data model.

Description

This operation returns the current properties for the custom group object designated by *{group-id}*.

The URI path *{group-id}* must designate an existing custom group object.

Authorization requirements

This operation has the following authorization requirement:
- Object-access permission to the custom group object designated by *{group-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 504.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The **object-id** in the URI (*{group-id}*) does not designate an existing custom group, or the API user does not have sufficient access (as described above).

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341 HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
```

Figure 249. Get Custom Group Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:45:45 GMT
content-type: application/json;charset=UTF-8
content-length: 250
{
   "class": "group",
   "description": "Spacely Sprockets Web Servers",
   "is-locked": false,
   "match-info": {},
   "name": "SS-Web-Servers",
   "object-id": "ee2782af-dd98-3ec0-bc2d-cfe2e9154341",
   "object-uri": "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341",
   "parent": null
}
```

Figure 250. Get Custom Group Properties: Response

Create Custom Group

Use the **Create Custom Group** operation to create a custom group name.

HTTP method and URI

POST /api/groups

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
name	String	Required	The name for the new custom group object
description	String	Optional	The description for the new custom group object
match-info	match-info object	Optional	A nested object describing the pattern match. If not provided, this is not a pattern-match custom group. Refer to "Class specific additional properties" on page 502 for details.

Response body contents

Field name	Type	Description
object-uri	String	The object URI of the new custom group.

Description

Group objects are programmatically identified by object-id and not by name. To avoid the confusion which might result from allowing redundant names, the **name** property is required for this operation, and the (case-sensitive) value supplied for the name property must be distinct from that of all currently-existing group objects. In keeping with restrictions imposed by the Hardware Management Console's Graphical User Interface (GUI), the following set of names is also not allowed:

- the current name of the Hardware Management Console
- the GUI View names {"Groups", "Exceptions", "Active Tasks", "Console Actions", "Task List", "Books", "Help", "Ensemble"}

On success, a custom group managed object is created reflecting the Request Body contents and HTTP status code 201 (Created) is returned.

Authorization requirements

This operation has the following authorization requirement:

- Action/task permission to the **Grouping** task.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents." In addition, the **Location** response header contains the URI of the newly created object.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	261	One of the following errors was detected: • The pattern string specified in **match-info** is not valid. This must be a non-empty string which is a valid regular expression. • One or more of the types specified in **match-info** is invalid. At least one type must be specified, and all must be values as documented for the **match-info types** property.
	290	The requested name is either reserved or already in use.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/groups HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
content-type: application/json
content-length: 74
{
   "description": "Spacely Sprockets Web Servers",
   "name": "SS-Web-Servers"
}
```

Figure 251. Create Custom Group: Request

```
201 Created
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:45:44 GMT
content-type: application/json;charset=UTF-8
content-length: 65
{
   "object-uri": "/api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341"
}
```

Figure 252. Create Custom Group: Response

Delete Custom Group

Use the **Delete Custom Group** operation to delete a custom group.

HTTP method and URI

DELETE /api/groups/{*group-id*}

In this request, the URI variable *{group-id}* is the object ID of the group.

Description

If successful, the custom group managed object designated by *{group-id}* is deleted.

If *{group-id}* does not identify an existing custom group, status code 404 (Not Found) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object-access permission to the custom group designated by *{group-id}*
- Action/task permission for the **Grouping** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
404 (Not Found)	1	The URI's *{group-id}* does not designate an existing custom group object, or the API user does not have object access to the group.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341 HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
```

Figure 253. Delete Custom Group: Request

```
204 No Content
date: Fri, 25 Nov 2011 16:45:45 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 254. Delete Custom Group: Response

Add Member to Custom Group

Use the **Add Member to Custom Group** operation to add a member to a custom group.

HTTP method and URI

POST /api/groups/{*group-id*}/operations/add-member

In this request, the URI variable *{group-id}* is the object ID of the group.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
object-uri	String	Required	The object URI of the object to be added to the group

Description

If successful, the managed object designated in the request body attains membership in the custom group identified by *{group-id}*.

The operation is subject to the following restrictions:
- The designated managed object must exist and must not already be a member of the group identified by *{group-id}*
- The group identified by *{group-id}* must be a custom group defined without a pattern-matching specification.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the custom group object designated by *{group-id}*
- Object access permission to the object designated by the request body
- Action/task permission for the **Grouping** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	291	The designated managed object is already a member of the custom group identified by *{group-id}*.
	294	The group identified by *{group-id}* was defined using pattern-matching.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
	293	The addition of the member to the custom group designated by the URI (*{group-id}*) would introduce a circular reference, which is not permitted.
404 (Not Found)	1	The URI's *{group-id}* does not designate an existing custom group object, or the API user does not have object access to the group.
	2	The request body does not designate an existing managed object, or the API user does not have sufficient access to the managed object.

HTTP error status code	Reason code	Description
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341/operations/add-member HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
content-type: application/json
content-length: 75
{
   "object-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af"
}
```

Figure 255. Add Member to Custom Group: Request

```
204 No Content
date: Fri, 25 Nov 2011 16:45:44 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 256. Add Member to Custom Group: Response

Remove Member from Custom Group

Use the **Remove Member from Custom Group** operation to remove a member from a custom group.

HTTP method and URI

POST /api/groups/{group-id}/operations/remove-member

In this request, the URI variable *{group-id}* is the object ID of the group

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
object-uri	String	Required	The object URI of the object to be removed from the group

Description

The managed object designated in the request body relinquishes its membership in the custom group identified by *{group-id}*.

The operation is subject to the following restrictions:
- The managed object designated in the request body must currently be a member of the group identified by *{group-id}*.

- The group identified by *{group-id}* must be a custom group defined without a pattern-matching specification

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the custom group object designated by *{group-id}*
- Object access permission to the object designated by the request body
- Action/task permission for the **Grouping** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	291	The designated managed object is not a member of the custom group identified by *{group-id}*.
	294	The group identified by *{group-id}* was defined using pattern-matching.
403 (Forbidden)	1	The user under which the API request was authenticated does not have the required authority to perform this operation.
404 (Not Found)	1	The URI's *{group-id}* does not designate an existing custom group object, or the API user does not have object access to the group.
	2	The request body does not designate an existing managed object, or the API user does not have sufficient access to the managed object.
500 (Server Error)	273	An unexpected error occurred during the operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
POST /api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341/operations/remove-member HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
content-type: application/json
content-length: 75
{
   "object-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af"
}
```

Figure 257. Remove Member from Custom Group: Request

```
204 No Content
date: Fri, 25 Nov 2011 16:45:45 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache

<No response body>
```

Figure 258. Remove Member from Custom Group: Response

List Custom Group Members

Use the **List Custom Group Members** operation to list custom group members.

HTTP method and URI

GET /api/groups/{*group-id*}/members

In this request, the URI variable *{group-id}* is the object ID of the group.

Response body contents

Field name	Type	Description
members	Array of nested objects	Array of nested objects which identify members of the custom group designated by *{group-id}*.

Each nested member object contains the following fields:

Field name	Type	Description
object-uri	String/URI	The value of the member object's **object-uri** property.
name	String	The value of the member object's **name** property.

Description

This operation lists the members of the custom group object designated by *{group-id}*. The results of this operation only include references to member objects for which the API user has object access authority.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section. If the custom group currently has no members, HTTP status code 200 (OK) is returned, along with an empty response body.

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the group object.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

On error, appropriate HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The custom group designated by the URI (*{group-id}*) does not exist, or the API user does not have sufficient access (as described above).

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/groups/ee2782af-dd98-3ec0-bc2d-cfe2e9154341/members HTTP/1.1
x-api-session: 42r6t4chltipvd6l4l61wi3111tf7fv2hes80hjqjs3invt7cp
```

Figure 259. List Custom Group Members: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:45:45 GMT
content-type: application/json;charset=UTF-8
content-length: 207
{
   "members": [
      {
         "name": "SS-Web-Svr-1",
         "object-uri": "/api/virtual-servers/576569dc-0db7-11e1-b1f1-f0def14b63af"
      },
      {
         "name": "SS-Web-Svr-2",
         "object-uri": "/api/virtual-servers/588d8c18-0db7-11e1-b1f1-f0def14b63af"
      }
   ]
}
```

Figure 260. List Custom Group Members: Response

CPC object

A CPC object represents a single zEnterprise Central Processor Complex (CPC) that is being managed by an HMC.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This object includes the properties defined in the "Base managed object properties schema" on page 33, with the following class-specific specialization:

Table 135. CPC object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path of the CPC object, of the form `/api/cpcs/{cpc-id}` where *{cpc-id}* is the value of the **object-id** property of the CPC object.
parent	—	String/ URI	If the CPC is a member of an ensemble, the parent is the canonical URI path for the ensemble object. Otherwise, this property contains a null object.
class	—	String	The class of a CPC object is **"cpc"**.
name	(ro)	String (1-8)	The CPC name
description	(ro)	String (0-1024)	The descriptive text associated with this CPC object.
status	(sc)	String Enum	The current operational status of the CPC object. One of: • **"operating"** - the CPC is operational • **"not-communicating"** - the CPC is not communicating with the HMC • **"exceptions"** - the CPC has one or more unusual conditions • **"status-check"** - the SE is not communicating with the CPC • **"service"** - the CPC has been placed in service mode • **"not-operating"** - the CPC is not operational • **"no power"** - the CPC has no power • **"service-required"** - the CPC is operating on the last redundant part of a particular type • **"degraded"** - one or more of the CPC elements are degraded.
acceptable-status	(w)(pc)	Array of String Enum	The set of operational **status** values in which the CPC object can exist and be considered in an acceptable (not alert causing) state. One or more of the values listed for the **status** property.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties.

Table 136. CPC object: class specific additional properties

Name	Qualifier	Type	Description
se-version	(pc)	String (1-8)	The current release level of the primary SE internal code. For example, “2.11.1”. Note that the alternate SE is normally at the same level, except when installing new internal code levels.
has-hardware-messages	(pc)	Boolean	The CPC object has hardware messages (true), or does not have hardware messages (false).
is-ensemble-member	(pc)	Boolean	Whether the CPC is currently part of an ensemble (true) or not (false).
iml-mode	(pc)	String Enum	The Initial Microcode Load (IML) mode type of the CPC object. One of: • **"not-set"** - the CPC is not IMLed • **"esa390"** - the CPE is in ESA/390 mode • **"lpar"** - the CPC is in logical partition mode • **"esa390-tpf"** - the CPC is in ESA/390 TPF mode
next-activation-profile-name	(w)(pc)	String (1-16)	The name of the activation profile to be used on the next activation of the CPC.
last-used-activation-profile-name	(pc)	String (0-16)	The name of the activation profile used on the last activation of the CPC or a null string.

Table 136. CPC object: class specific additional properties (continued)

Name	Qualifier	Type	Description
machine-model	(pc)	String (1-3)	The model of the machine containing this CPC. For example, “M15”.
machine-type	(pc)	String (1-4)	The type of the machine containing this CPC. For example, “2817”.
machine-serial-number	(pc)	String (1-12)	The serial number of the machine containing this CPC. For example, “00 - SP1D92B”.
cpc-serial-number	—	String (1-12)	The serial number of the CPC. For example, “00000SP1D92B”.
cpc-node-descriptor	(pc)	String (2)	The hexadecimal number mapped to the device location of the CPC. This property identifies the CPC's relative order among other CPCs, if any, in the machine. For example, “00”.
is-cbu-installed	—	Boolean	The Capacity Backup Upgrade (CBU) facility is installed (true), or not installed (false). Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
is-cbu-enabled	—	Boolean	CBU is enabled (true), or is not enabled (false). Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
is-cbu-activated	—	Boolean	CBU is activated (true), or is not activated (false). Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
is-real-cbu-available	—	Boolean	Real CBU is available (true), or not available (false). Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
cbu-activation-date	—	Timestamp	The date of CBU activation or a null string. Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
cbu-expiration-date	—	Timestamp	The date of CBU expiration or a null string. Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
cbu-number-of-tests-left	—	Integer	The number of CBU tests left. Note: if status is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User's Guide* for details.
is-service-required	—	Boolean	Whether the CPC is operating using the last redundant part of a particular type (true) or not (false). If true, repairs should be made before additional parts fail that would make this CPC non-operational. Refer to the *Support Element Operations Guide* for more details.
degraded-status	(pc)	Array of String Enum	The set of degraded status values. If the CPC is not degraded, this property contains “not-degraded” as the only value. Otherwise, this property contains one or more of the following: • **"memory"** - loss of memory • **"io"** - loss f I/O channels • **"node"** - one or more books are no longer functioning • **"ring"** - the ring connecting the book is open • **"cbu"** - CBU resources have expired • **"mru"** - cycle time reduction due to an MRU problem • **"ambient-temp"** -cycle time reduction due to a temperature problem • **"iml"** - CPC was IMLed during cycle time reduction.

Table 136. CPC object: class specific additional properties (continued)

Name	Qualifier	Type	Description
processor-running-time-type	(w)	String Enum	Denotes how the processor-running-time property value was determined. One of: • **"system-determined"** - the processor running time is dynamically determined by the system • **"user-determined"** - the processor running time is set to a constant value. Note: if **iml-mode** is not **"lpar"**, a null object is returned.
processor-running-time	(w)	Integer	The amount of continuous time, in milliseconds, allowed for logical processors to perform jobs on shared processors for the CPC object. Note: a null object is returned if the **iml-mode** is not **"lpar"** or processor-running-time-type is **"system-determined"**.
does-wait-state-end-time-slice	(w)	Boolean	Logical Partitions of the CPC should lose their share of running time when they enter a wait state (true), or should not lose their share of running time when they enter a wait state (false). Note: a null object is returned if the **iml-mode** is not **"lpar"** or processor-running-time-type is **"system-determined"**.
is-on-off-cod-installed	—	Boolean	On/Off Capacity on Demand is installed for the CPC object (true), or is not installed (false). Note: if **status** is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User’s Guide* for details.
is-on-off-cod-enabled	—	Boolean	On/Off CoD is enabled (true), or is not enabled (false). Note: if **status** is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User’s Guide* for details.
is-on-off-cod-activated	—	Boolean	On/Off CoD is currently activated for the CPC object (true), or is not currently activated (false). Note: if **status** is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User’s Guide* for details.
on-off-cod-activation-mode	—	Timestamp	The date when On/Off CoD was activated. Note: if **status** is “not-communicating”, a null object is returned. Refer to the *Capacity On Demand User’s Guide* for details.
software-model-permanent	(pc)	String (1-3)	The software model based on the permanent processors. For example, “700”.
software-model-permanent-plus-billable	(pc)	String (1-3)	The software model based on the permanent plus billable processors. For example, “700”.
software-model-permanent-plus-temporary	(pc)	String (1-3)	The software model based on the permanent plus all temporary processors. For example, “700”.
msu-permanent	—	Integer	The MSU value associated with the software model based on the permanent processors.
msu-permanent-plus-billable	—	Integer	The MSU value associated with the software model based on the permanent plus billable processors.
msu-permanent-plus-temporary	—	Integer	The MSU value associated with the software model based on the permanent plus all temporary processors.
processor-count-general-purpose	—	Integer	The count of active general purpose processors.
processor-count-service-assist	—	Integer	The count of active service assist processors.

Table 136. CPC object: class specific additional properties (continued)

Name	Qualifier	Type	Description
processor-count-aap	—	Integer	The count of active IBM System z Application Assist Processor (zAAP) processors.
processor-count-ifl	—	Integer	The count of active IBM System z Integrated Facility for Linux (IFL) processors.
processor-count-icf	—	Integer	The count of active Internal Coupling Facility (ICF) processors.
processor-count-iip	—	Integer	The count of active IBM System z Integrated Information Processor (zIIP) processors.
processor-count-defective	—	Integer	The count of defective processors. Includes all processor types.
processor-count-spare	—	Integer	The count of spare processors. Includes all processor types.
processor-count-pending	—	Integer	The combined number of processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records.
processor-count-pending-general-purpose	—	Integer	The number of general purpose processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
processor-count-pending-service-assist	—	Integer	The number of service assist processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
processor-count-pending-aap	—	Integer	The number of Application Assist processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
processor-count-pending-ifl	—	Integer	The number of Integrated Facility for Linux processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
processor-count-pending-icf	—	Integer	The number of Integrated Coupling Facility processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
processor-count-pending-iip	—	Integer	The number of Integrated Information processors that will become active, when more processors are made available by adding new hardware or by deactivating capacity records. Note: if status is “not-communicating”, a null object is returned.
has-temporary-capacity-change-allowed	—	Boolean	Whether API applications are allowed to make changes to temporary capacity (true), or not (false).
network-info	(pc)	network-info object	A nested object that details the network information for this CPC.
ec-mcl-description	—	ec-mcl-description object	Describes the Engineering Change (EC) and MicroCode Level (MCL) for the CPC object. An empty object is returned if the information is unavailable from the SE. Refer to the description of the `ec-mcl-description` object for details.

Table 136. CPC object: class specific additional properties (continued)

Name	Qualifier	Type	Description
has-automatic-se-switch-enabled	—	Boolean	Automatic switching between primary and alternate Support Elements is enabled for the CPC object (true), or is not enabled (false). Refer to the *Support Element Operations Guide* for details on automatic SE switching.
stp-configuration	—	stp-config object	Describes the Server Time Protocol (STP) configuration. Refer to the description of the `stp-config` object for details. Note: if the required feature(s) are not installed, the property is not returned.
lan-interface1-type	(pc)	String Enum	The adapter type of the Support Element's LAN interface 1. One of the following: • **"ethernet"** • **"token-ring"** • **"unknown"**
lan-interface1-address	(pc)	String (1-2)	The MAC address of the Support Element's LAN interface 1.
lan-inerface2-type	(pc)	String Enum	The adapter type of the Support Element's LAN interface 2. One of the following: • **"ethernet"** • **"token-ring"** • **"unknown"**
lan-interface2-address	(pc)	String (1-12)	The MAC address of the Support Element's LAN interface 2.
network1-ipv4-mask	(pc)	String (1-15)	The network IP mask value.
network2-ipv4-pri-ipaddr	(pc)	String IPV4 address	The primary IPv4 address or a null object if not configured.
network1-ipv4-alt-ipaddr	(pc)	String IPV4 address	The alternate IPv4 address or a null object if not configured.
network1-ipv6-info	—	Array of ipv6-info objects	A list of objects describing the Support Element's IPv6 network connections. If no IPv6 connections are defined, an empty list is returned.
network2-ipv4-mask	(pc)	String (1-15)	The network IP mask value.
network2-ipv4-pri-ipaddr	(pc)	String IPCV4 address	The primary IPv4 address or a null object if not configured.
network2-ipv4-alt-ipaddr	(pc)	String IPV4 address	The alternate IPv4 address or a null object if not configured.
network2-ipv6-info	—	Array of ipv6-info objects	A list of objects describing the Support Element's IPV6 network connections. If no IPv6 connections are defined, an empty list is returned.

Table 137. ipv6-info object properties

Name	Type	Description
type	String Enum	The IPv6 scope. One of the following values: • **"link-local"** • **"static"** • **"auto"**
prefix	Integer	The number of leading bits of the IPv6 address that represent the network prefix
pri-ip-address	String IPV6 address	The primary IPv6 address

Table 137. ipv6-info object properties (continued)

Name	Type	Description
alt-ip-address	String IPV6 address	The alternate IPv6 address or a null object if not configured

Energy management related additional properties

In addition to the properties defined above, this object includes the following additional class-specific properties related to energy management. For further explanation of the various states involved, please see "Special states" on page 138.

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

Table 138. CPC object: energy management related additional properties

Name	Qualifier	Type	Description
cpc-power-rating	—	Integer	Specifies the maximum power draw in watts (W) of this CPC. This is a calculated value as indicated by the electrical rating labels or system rating plates of the CPC components.
cpc-power-consumption	(mg)	Integer	Specifies the current power consumption in watts (W) for this CPC. The CPC power consumption includes the power consumption of the zCPC and BladeCenters. The BladeCenter power consumption includes the power consumption of the blades contained within the BladeCenter. If the system does not include a BladeCenter, the CPC power consumption will be equal to the zCPC power consumption.
cpc-power-saving	—	String Enum	Specifies the current power saving setting of the CPC. Power saving is used to reduce the energy consumption of a system and can be managed in the Set Power Saving operation. The possible settings include: • **"high-performance"** - The power consumption and performance of the CPC are not reduced. This is the default setting. • **"low-power"** - All components of the CPC enabled for power saving will have reduced performance to allow for low power consumption. • **"custom"** - Custom mode indicates that some, but not all, components of the CPC are in the Low power setting. • **"not-supported"** - Power saving is not supported for this CPC. • **"not-available"** - Specifies that **cpc-power-saving** property could not be read from this CPC. • **"not-entitled"** - The server is not entitled for Power saving.

Table 138. CPC object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
cpc-power-saving-state	—	String Enum	Specifies the power saving setting of the CPC set by the user. Please note that this property indicates the user setting and may not match the real state of the hardware compared to the **cpc-power-saving** property. For more information, see "Group power saving" on page 139. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption and performance of the CPC. • **"low-power"** - Specifies low power consumption for all components of the CPC enabled for power saving. • **"custom"** - Specifies that the CPC does not control the children. This is the default setting. • **"not-supported"** - Specifies that power saving is not supported for this CPC. • **"not-entitled"** - Specifies that the server is not entitled to power saving.
cpc-power-save-allowed	—	String Enum	Should be used to determine if a call of the power save operation is currently allowed. If a value other that **"allowed"** is returned the caller may reckon that the power save operation will fail. The possible settings include: • **"allowed"** - Alter power save setting is allowed for this CPC • **"unknown"** - Unknown reason • **"not-supported"** - Power saving is not supported for this CPC. • **"not-entitled"** - Specifies the server is not entitled to power capping.
cpc-power-capping-state	—	String Enum	Specifies the current power capping setting of the CPC. Power capping is used to limit peak power consumption of a system and can be managed in the Set Power Cap operation. The possible settings include: • **"disabled"** - The power cap of the CPC is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - All components of the CPC available for power capping will be capped to limit the peak power consumption of the CPC. • **"custom"** - The components of the CPC can be individually configured for power capping. • **"not-supported"** - Power capping is not supported for this CPC. • **"not-entitled"** - The server is not entitled for Power capping.
cpc-power-cap-minimum	—	Integer	Specifies the minimum value for the CPC cap value in watts (W). This is a sum of the component minimum cap values.
cpc-power-cap-maximum	—	Integer	Specifies the maximum value for the CPC cap value in watts (W). This is a sum of the component maximum cap values.
cpc-power-cap-current	—	Integer	Specifies the current cap value for the CPC in watts (W). The current cap value indicates the power budget for the CPC and is the sum of the component Cap values.

Table 138. CPC object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
cpc-power-cap-allowed	—	String Enum	Should be used to determine if a call of the power capping operation is currently allowed. If a value other that **"allowed"** is returned the caller may reckon that the power capping operation will fail. The possible settings include: • **"allowed"**- Alter power capping setting is allowed for this CPC • **"unknown"** - Unknown reason • **"not-supported"** - Power capping is not supported for this CPC. • **"not-entitled"** - Specifies the server is not entitled to power capping.
zcpc-power-rating	—	Integer	Specifies the maximum power draw in watts (W) of this zCPC. This is a calculated value as indicated by the electrical rating labels or system rating plates of the zCPC components.
zcpc-power-consumption	(mg)	Integer	Specifies the current power consumption of the zCPC in watts (W).
zcpc-power-saving	—	String Enum	Specifies the current power saving setting of the zCPC. Power saving is used to reduce the energy consumption of a system and can be managed in the Set Power Saving operation. The possible settings include: • **"high-performance"** - The power consumption and performance of the zCPC are not reduced. This is the default setting. • **"low-power"** - The performance of the zCPC is reduced to allow for low power consumption. • **"not-supported"** - Power saving is not supported for this zCPC. • **"not-available"** - Specifies that **zcpc-power-saving** property could not be read for this zCPC. • **"not-entitled"** - The server is not entitled for Power saving.
zcpc-power-saving-state	—	String Enum	Specifies the power saving setting of the zCPC set by the user. Please note that this property indicates the user setting and may not match the real state of the hardware compared to the **zcpc-power-saving** property. For more information, see "Group power saving" on page 139. The possible settings include: • **"high-performance"** - Specifies not reducing the power consumption and performance of the zCPC. This is the default setting. • **"low-power"** - Specifies low power consumption for all components of the zCPC enabled for power saving. • **"not-supported"** - Specifies that power saving is not supported for this zCPC. • **"not-entitled"** - Specifies that the server is not entitled to power saving.

Table 138. CPC object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
zcpc-power-save-allowed	—	String Enum	Should be used to determine if a call of the power save operation is currently allowed. If a value other that **"allowed"** is returned the caller may reckon that the power save operation will fail. The possible settings include: • **"allowed"** - Alter power save is allowed for this zCPC • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power save. • **"under-group-control"** - The zCPC is under group control and cannot be individually altered. • **"not-supported"** - Power saving is not supported for this zCPC. • **"once-a-day-exceeded"** - Power saving mode has been entered at some point during the day and will not be allowed again until the next calendar day.
zcpc-power-capping-state		String Enum	Specifies the current power capping setting of the zCPC. Power capping is used to limit peak power consumption of a system and can be managed in the Set Power Cap operation. The possible settings include: • **"disabled"** - The power cap of the zCPC is not set and the peak power consumption is not limited. This is the default setting. • **"enabled"** - The peak power consumption of the zCPC is limited to the Current cap value. • **"custom"** - The components of the CPC can be individually configured for power capping. • **"not-supported"** - Power capping is not supported for this zCPC. • **"not-entitled"** - The server is not entitled for Power capping.
zcpc-power-cap-minimum	—	Integer	Specifies the minimum value for the zCPC cap value in watts (W).
zcpc-power-cap-maximum	—	Integer	Specifies the maximum value for the zCPC cap value in watts (W).
zcpc-power-cap-current	—	Integer	Specifies the current cap value for the CPC in watts (W). The current cap value indicates the power budget for the zCPC.
zcpc-power-cap-allowed	—	String Enum	Should be used to determine if a call of the power capping operation is currently allowed. If a value other that **"allowed"** is returned the caller may reckon that the power capping operation will fail. The possible settings include: • **"allowed"** - Alter power capping is allowed for this zCPC • **"unknown"** - Unknown reason • **"not-entitled"** - Specifies the server is not entitled to power save. • **"not-supported"** - Power capping is not supported for this zCPC. • **"under-group-control"** - Power capping is under group control
zcpc-ambient-temperature	(mg)	Float	Specifies the input air temperature in degrees Celsius (ºC) as measured by the system.
zcpc-exhaust-temperature	—	Float	Specifies the exhaust air temperature in degrees Celsius (ºC) as calculated by the system. This is useful in determining potential hot spots in the data center.

Table 138. CPC object: energy management related additional properties (continued)

Name	Qualifier	Type	Description
zcpc-humidity	(mg)	Integer	Specifies the amount of water vapor in the air as measured by the system. The humidity sensor gives a reading of the relative humidity of the air entering the system. The recommended long-term relative humidity for a system with an altitude from sea level to 900 meters (2953 feet) is 60%. The range of acceptable relative humidity is 8% - 80%. For more information, see the chapter related to environmental specifications in the *Installation Manual for Physical Planning*.
zcpc-dew-point	(mg)	Float	Specifies the air temperature in degrees Celsius (°C) at which water vapor will condense into water. This is a calculated value based on the current temperature and relative humidity. Cooling the server to the dew point can result in condensation on critical internal parts, leading to equipment failure, unless the computer room environment is adequately maintained to prevent it. For more information, see the chapter related to environmental specifications in the *Installation Manual for Physical Planning*.
zcpc-heat-load	(mg)	Integer	Specifies the amount of heat in Btu/hr. removed from the system.
zcpc-heat-load-forced-air	—	Integer	Specifies the amount of heat in Btu/hr. removed from the system by forced-air.
zcpc-heat-load-water	—	Integer	Specifies the amount of heat in Btu/hr. removed from the system by chilled water. The value is always 0 on an air cooled system.
zcpc-maximum-potential-power	—	Integer	Specifies the maximum potential power consumption of a system in watts (W). This value is based on the configuration of the system and can be used for power and cooling planning.
zcpc-maximum-potential-heat-load	—	Integer	Specifies the maximum potential heat load of a system in Btu/hr. This value is based on the configuration of the system and can be used for power and cooling planning.

List CPC Objects

The **List CPC Objects** operation returns a list of the zManager Web Services API capable CPCs managed by an HMC.

HTTP method and URI

```
GET /api/cpcs
```

Query Parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
cpcs	Array of cpc-info objects	Array of nested cpc-info objects (described in the next table). If no matching CPC objects are found, an empty array is returned.

Each nested cpc-info object contains the following fields:

Field name	Type	Description
object-uri	String/URI	Canonical URI path of the CPC object
name	String	The name of the CPC object
status	String Enum	The current status of the CPC object

Description

This operation lists the zManager Web Services API capable CPC objects that are managed by this HMC. The object URI, object ID and display name are provided for each CPC returned. CPCs that are not zManager Web Services API capable are not returned.

If the name query parameter is specified, the returned list is limited to those CPC objects that have a name property matching the specified filter pattern. If the name parameter is omitted, this filtering is not done.

An object is only included in the list if the API user has object-access permission for that object.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to any CPC object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs HTTP/1.1
x-api-session: 2jm2h7j25d1e1g5wbygmfriyjiit8tp4iqiw8h09j8kz68i0k6
```

Figure 261. List CPC Objects: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 07:18:42 GMT
content-type: application/json;charset=UTF-8
content-length: 492
{
   "cpcs": [
      {
         "name": "P0LXSMOZ",
         "object-uri": "/api/cpcs/e8753ff5-8ea6-35d9-b047-83c2624ba8da",
         "status": "not-operating"
      },
      {
         "name": "R32",
         "object-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
         "status": "operating"
      },
      {
         "name": "ICHABOD",
         "object-uri": "/api/cpcs/ac15c987-90c6-3526-854e-4c612939260d",
         "status": "not-operating"
      }
   ]
}
```

Figure 262. List CPC Objects: Response

List Ensemble CPC Objects

The **List Ensemble CPC Objects** operation returns a list of the CPCs associated with an ensemble.

HTTP method and URI

GET /api/ensembles/{*ensemble-id*}/cpcs

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`cpcs`	Array of cpc-info objects	Array of nested cpc-info objects (described in the next table). If no matching CPC objects are found, an empty array is returned.

Each nested cpc-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the CPC object
`name`	String	The name of the CPC object
`status`	String Enum	The current status of the CPC object

Description

This operation lists the CPC objects that are associated with the ensemble identified in the request URI. The object URI, object ID and display name are provided for each CPC returned. In contrast, the List Ensemble Nodes operation lists objects of any type that are associated with an ensemble.

If the **name** query parameter is specified, the returned list is limited to those CPC objects that have a **name** property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

A CPC object is only included in the list if the API user has object-access permission for that CPC object.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the ensemble object designated by *{ensemble-id}*
- Object access permission to any CPC object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID in the URI (*{ensemble-id}*) does not designate an existing ensemble object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334/cpcs HTTP/1.1
x-api-session: 2jm2h7j25d1e1g5wbygmfriyjiit8tp4iqiw8h09j8kz68i0k6
```

Figure 263. List Ensemble CPC Objects: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 07:18:42 GMT
content-type: application/json;charset=UTF-8
content-length: 214
{
   "cpcs": [
      {
         "name": "R32",
         "object-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
         "status": "operating"
      },
      {
         "name": "ICHABOD",
         "object-uri": "/api/cpcs/ac15c987-90c6-3526-854e-4c612939260d",
         "status": "not-operating"
      }
   ]
}
```

Figure 264. List Ensemble CPC Objects: Response

Get CPC Properties

The **Get CPC Properties** operation retrieves the properties of a single CPC object designated by *{cpc-id}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Response body contents

On successful completion, the response body provides the current values of the properties for the CPC object as defined in "Data model" on page 513.

Description

Some CPC properties are only available if the HMC is communicating with the SE, and are returned as null objects if the HMC is not communicating with the SE.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined "Data model" on page 513.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 527.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	272	Unable to obtain Server Time Protocol (STP) configuration. Retry the request later.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340 HTTP/1.1
x-api-session: 65aw2jahugn1wop51hsq0c6aldkkx773dz9ulirrvg2z853m4u
```

Figure 265. Get CPC Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:58:36 GMT
content-type: application/json;charset=UTF-8
content-length: 9001
{
   "acceptable-status": [
      "operating"
   ],
   "additional-status": "",
   "cbu-activation-date": 0,
   "cbu-expiration-date": 0,
   "cbu-number-of-tests-left": 0,
   "class": "cpc",
   "cpc-node-descriptor": "00",
   "cpc-power-cap-allowed": "allowed",
   "cpc-power-cap-current": 90000,
   "cpc-power-cap-maximum": 106936,
   "cpc-power-cap-minimum": 16019,
   "cpc-power-capping-state": "enabled",
   "cpc-power-consumption": 8160,
   "cpc-power-rating": 46522,
   "cpc-power-save-allowed": "allowed",
   "cpc-power-saving": "high-performance",
   "cpc-serial-number": "000020076D25",
   "degraded-status": [
      "not-degraded"
   ],
   "description": "Central Processing Complex (CPC)",
   "does-wait-state-end-time-slice": null,
   "ec-mcl-description": {
      "action": [
         {
            "activation": "current",
            "pending": false,
            "type": "channel-config"
         },
         {
            "activation": "current",
            "pending": false,
            "type": "coupling-facility-reactivation"
         },
         {
            "activation": "current",
            "pending": false,
            "type": "power-on-reset-tracking"
         },
         {
            "activation": "current",
            "pending": true,
            "type": "zhybrid-blades-activation"
         },
         {
            "activation": "next",
            "pending": false,
            "type": "channel-config"
         },
         {
```

Figure 266. Get CPC Properties: Response (Part 1)

```
            "activation": "next",
            "pending": false,
            "type": "coupling-facility-reactivation"
         },
         {
            "activation": "next",
            "pending": false,
            "type": "power-on-reset-tracking"
         },
         {
            "activation": "next",
            "pending": false,
            "type": "zhybrid-blades-activation"
         }
      ],
      "ec": [
         {
            "description": "SE Framework",
            "mcl": [
               {
                  "last-update": 1321019308748,
                  "level": "216",
                  "type": "retrieved"
               },
               {
                  "last-update": 1321019499749,
                  "level": "216",
                  "type": "activated"
               },
               {
                  "last-update": 1320675688749,
                  "level": "179",
                  "type": "accepted"
               },
               {
                  "last-update": 943938000748,
                  "level": "216",
                  "type": "installable-concurrent"
               },
               {
                  "last-update": 943938000749,
                  "level": "180",
                  "type": "removable-concurrent"
               }
            ],
            "number": "N48168",
            "part-number": "45D8918",
            "type": "Base EC"
         },
         {
            "description": "IBM x86 Blade Concurrent Components",
            "mcl": [
               {
                  "last-update": 1321019296651,
                  "level": "022",
                  "type": "retrieved"
               },
               {
```

Figure 267. Get CPC Properties: Response (Part 2)

```
                "last-update": 1320675685652,
                 "level": "009",
                 "type": "accepted"
              },
              {
                 "last-update": 943938000652,
                 "level": "022",
                 "type": "installable-concurrent"
              },
              {
                 "last-update": 943938000652,
                 "level": "010",
                 "type": "removable-concurrent"
              }
           ],
           "number": "N48140",
           "part-number": "41U8008",
           "type": "Other Optional EC"
        },
        {
           "description": "Embedded Operating System T5xx Series",
           "mcl": [
              {
                 "last-update": 1321018125779,
                 "level": "001",
                 "type": "retrieved"
              },
              {
                 "last-update": 1321018167779,
                 "level": "001",
                 "type": "activated"
              },
              {
                 "last-update": null,
                 "level": "000",
                 "type": "accepted"
              },
              {
                 "last-update": 943938000779,
                 "level": "001",
                 "type": "installable-concurrent"
              },
              {
                 "last-update": 943938000779,
                 "level": "001",
                 "type": "removable-concurrent"
              }
           ],
           "number": "N48197",
           "part-number": "45D8919",
           "type": "Base EC"
        }
     ]
  },
```

Figure 268. Get CPC Properties: Response (Part 3)

```
"has-automatic-se-switch-enabled": true,
"has-hardware-messages": true,
"has-temporary-capacity-change-allowed": false,
"has-unacceptable-status": false,
"iml-mode": "lpar",
"is-cbu-activated": false,
"is-cbu-enabled": true,
"is-cbu-installed": false,
"is-ensemble-member": true,
"is-locked": false,
"is-on-off-cod-activated": false,
"is-on-off-cod-enabled": true,
"is-on-off-cod-installed": false,
"is-real-cbu-available": false,
"is-service-required": false,
"lan-interface1-address": "f0def14b63af",
"lan-interface1-type": "ethernet",
"lan-interface2-address": "f0def14b63af",
"lan-interface2-type": "ethernet",
"last-used-activation-profile-name": "DEFAULT         ",
"machine-model": "M15",
"machine-serial-number": "000020076D25",
"machine-type": "2817",
"msu-permanent": 1091,
"msu-permanent-plus-billable": 1091,
"msu-permanent-plus-temporary": 1091,
"name": "R32",
"network1-ipv4-alt-ipaddr": "9.60.15.6",
"network1-ipv4-mask": "255.255.255.0",
"network1-ipv4-pri-ipaddr": "9.60.15.5",
"network1-ipv6-info": [
   {
      "alt-ip-address": "fdd8:673b:d89b:1:f2de:f1ff:fe52:d359",
      "prefix": 64,
      "pri-ip-address": "fdd8:673b:d89b:1:f2de:f1ff:fe4b:63af",
      "type": "auto"
   },
   {
      "alt-ip-address": "fe80::f2de:f1ff:fe52:d359%eth0",
      "prefix": 64,
      "pri-ip-address": "fe80::f2de:f1ff:fe4b:63af%eth0",
      "type": "link-local"
   }
],
"network2-ipv4-alt-ipaddr": "9.60.14.6",
"network2-ipv4-mask": "255.255.255.0",
"network2-ipv4-pri-ipaddr": "9.60.14.5",
"network2-ipv6-info": [
   {
      "alt-ip-address": "fe80::f2de:f1ff:fe52:d359%eth1",
      "prefix": 64,
      "pri-ip-address": "fe80::f2de:f1ff:fe4b:63af%eth1",
      "type": "link-local"
   }
],
```

Figure 269. Get CPC Properties: Response (Part 4)

```
   "next-activation-profile-name": "TESTCDU",
   "object-id": "37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "object-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "on-off-cod-activation-date": 0,
   "parent": "/api/ensembles/f8fc3a9c-03f2-11e1-ba83-0010184c8334",
   "processor-count-aap": 1,
   "processor-count-defective": 0,
   "processor-count-general-purpose": 9,
   "processor-count-icf": 2,
   "processor-count-ifl": 2,
   "processor-count-iip": 1,
   "processor-count-pending": 0,
   "processor-count-pending-aap": 0,
   "processor-count-pending-general-purpose": 0,
   "processor-count-pending-icf": 0,
   "processor-count-pending-ifl": 0,
   "processor-count-pending-iip": 0,
   "processor-count-pending-service-assist": 0,
   "processor-count-service-assist": 3,
   "processor-count-spare": 0,
   "processor-running-time": null,
   "processor-running-time-type": "system-determined",
   "se-version": "2.11.1",
   "software-model-permanent": "709",
   "software-model-permanent-plus-billable": "709",
   "software-model-permanent-plus-temporary": "709",
   "status": "operating",
   "stp-configuration": {
      "current-time-server": "preferred",
      "etr-id": null,
      "stp-id": "12345678"
   },
   "zcpc-ambient-temperature": 25.399999618530273,
   "zcpc-dew-point": 2.4000000953674316,
   "zcpc-exhaust-temperature": 34.0,
   "zcpc-heat-load": 20293,
   "zcpc-heat-load-forced-air": 20293,
   "zcpc-heat-load-water": 0,
   "zcpc-humidity": 22,
   "zcpc-maximum-potential-heat-load": 26571,
   "zcpc-maximum-potential-power": 7782,
   "zcpc-power-cap-allowed": "under-group-control",
   "zcpc-power-cap-current": 23745,
   "zcpc-power-cap-maximum": 27400,
   "zcpc-power-cap-minimum": 7782,
   "zcpc-power-capping-state": "enabled",
   "zcpc-power-consumption": 5943,
   "zcpc-power-rating": 27400,
   "zcpc-power-save-allowed": "under-group-control",
   "zcpc-power-saving": "high-performance"
}
```


Figure 270. Get CPC Properties: Response (Part 5)

Update CPC Properties

The **Update CPC Properties** operation updates one or more writeable properties of the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain one or more field names representing writable CPC properties, along with the new values for those fields.

The request body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the CPC object type to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the object is updated with the value provided by the input field, and status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **CPC Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	268	The requested update requires that the **processor-running-time-type** property already contain **"user-determined"** or that the request body also requests an update of the **processor-running-time-type** property to **"user-determined"**.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Activate CPC

The **Activate CPC** operation activates the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/activate

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`activation-profile-name`	String (1-16)	Optional	The name of the activation profile to be used for the request. If not provided, the request uses the profile name specified in the activation-profile-name property for the CPC object.
`force`	Boolean	Optional	Whether this operation is permitted when the CPC is in **"operating"** status (true) or not (false). The default is false.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String/URI	URI that may be queried to retrieve activation status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in "Description." The **job-results** field is null for this operation.

Description

Activation is a process that makes a CPC operational, which means either:
- The CPC is ready to have a control program or operating system loaded, or
- The CPC has loaded and is running a control program or operating system.

Activation makes a CPC operational by:
- Using predefined information, referred to as an activation profile, to set the operational capabilities and characteristics of the CPC
- Checking the current status of the CPC, and then performing only the operations necessary to make it operational as specified in the activation profile.

So, using activation is not limited to starting the system. Using activation is recommended whenever you want to make the CPC or its logical partitions operational.

A complete activation activates the CPC and its logical partitions completely in a single step. The result of a complete activation is an operational CPC with logical partitions loaded and running operating systems. The current status of the CPC and its logical partitions determines which operations are performed during activation to make them operational. Activation may include:

1. Turning CPC power on.
2. Performing a power-on reset, this includes allocating system resources to the CPC.
3. Then activating logical partitions to support multiple images. Activating each logical partition includes:
   a. Initializing it.
   b. Allocating system resources to it.
   c. Loading it with a control program or operating system.

Because the status of the CPC and its logical partitions determines which operations must be performed during activation to make them operational, one or more operations listed above may not be performed during activation. For example:

- Activating the CPC does not perform a power-on reset if the CPC has already been power-on reset and the applicable settings in its assigned activation profile, such as the operating mode and active input/output configuration data set (IOCDS), are already in effect.
- Activating the CPC does not perform any operations if the CPC is already operational and all settings in its assigned activation profile are already in effect.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent, with Job Status and Reason Codes described in "Job status and reason codes" on page 537.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **Activate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 535.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status code	Job reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current CPC status and use of the force=false parameter. If rejected due to force=false, the CPC status is unchanged. If the operation failed, the CPC status is unknown. Refer to the message parameter in the error response body for details.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Deactivate CPC

The **Deactivate CPC** operation deactivates the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/deactivate

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
force	Boolean	Optional	Whether this operation is permitted when the CPC is in **"operating"** status (true) or not (false). The default is false.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve activation status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in "Job status and reason codes" on page 538. The **job-results** field is null for this operation.

Description

Deactivation is an orderly process for shutting down and turning off the CPC.

Shutting down and turning off the CPC, referred to also as deactivating the CPC, includes:
- Ending hardware and software activity
- Clearing, releasing, and de-allocating hardware resources
- Turning off power.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent, with Job Status and Reason Codes seen in "Job status and reason codes."

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **Deactivate** task.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 537.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

Job status code	Job reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current CPC status and use of the force=false parameter. If rejected due to force=false, the CPC status is unchanged. If the operation failed, the CPC status is unknown. Refer to the message parameter in the error response body for details.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Import Profiles

The **Import Profiles** operation imports activation profiles and/or system activity profiles for the CPC from the SE hard drive into the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/import-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
profile-area	Integer (1-4)	Required	The numbered hard drive area from which the profiles are imported. Use the profile-area value specified on the prior **Export Profiles** operation.

Description

The Support Element provides four reusable areas on its hard drive from which the data save by a prior Export Profiles can be read.

Exporting and importing profiles is necessary only when you intend to have your current system and Support Element replaced with a new system and Support Element. Refer to the *Support Element Operations Guide* for more details.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the CIM Actions ExportSettingsData task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.

HTTP error status code	Reason code	Description
500 (Server Error)	279	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Export Profiles

The **Export Profiles** operation exports activation profiles and/or system activity profiles from the CPC object designated by *{cpc-id}* to the SE hard drive.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/export-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`profile-area`	Integer (1-4)	Required	The numbered hard drive area to which the profiles are exported. Any existing data is overwritten.

Description

The Support Element provides four reusable areas on its hard drive that can be used as temporary save areas. The choice of save area is up to the caller.

Exporting and importing profiles is necessary only when you intend to have your current system and Support Element replaced with a new system and Support Element. Refer to the *Support Element Operations Guide* for more details.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the CIM Actions ExportSettingsData task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	279	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Add Temporary Capacity

The **Add Temporary Capacity** operation adds temporary processors or increases temporary model capacity to the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/add-temp-capacity

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`record-id`	String (1-8)	Required	Identifies the capacity record to be used for this request
`software-model`	String (1-3)	Optional	The target software model. This value must be one of the software models defined within the capacity record. Implicit in this value is the number of general processors added. If not provided, the current software model is not changed.
`processor-info`	Array of processor-info objects	Optional	A nested object that defines the number of specialty processors to be added. If not provided, the number of specialty processors is not changed.
`force`	Boolean	Optional	Whether the operation proceeds if not enough processors are available (true) or not (false). The default is false.
`test`	Boolean	Required	Whether the request should activate the changes (false) or not (true)

processor-info object

Field name	Type	Rqd/Opt	Description
`processor-type`	String Enum	Required	Identifies the type of specialty processors to be affected. One of: • **"aap"** - Application Assist Processor • **"ifl"** - Integrated Facility for Linux processor • **"icf"** - Internal Coupling Facility processor • **"iip"** - Integrated Information Processors • **"sap"** - System Assist Processor
`num-processor-steps`	Integer	Optional	The delta to the current number of processors. If not provided, the number of processors is not changed.

Description

Removal of these temporary resources can be performed manually via the Remove Temporary Capacity operation or automatically upon expiration of the capacity record.

Refer to the *Capacity on Demand User's Guide* for details on temporary capacity changes.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the CIM Actions Activate task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	271	A duplicate processor-type entry was found in the processor-info array, remove the duplicate entry.
	275	The test value does not match the value stored in the capacity record.
	276	Either the request specifies more resources than available or the requested software model specifies fewer resources than the current software model.
	277	A temporary capacity record is already active. It must be deactivated before a new capacity record can be activated.
	278	The software-model value was not found in the capacity record. Only software models as defined in the target capacity record can be specified.
	298	The operation parameters conflict with the capacity record type: • force=true is permitted only for CBU, CPE and loaner capacity records.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	274	The requested capacity record does not exist.

HTTP error status code	Reason code	Description
409 (Conflict)	1	The operation is unavailable in the current CPC state: • The SE is not configured to allow temporary capacity changes via an API • The CPC **status** property is not **"operating"** or the CPC iml-mode property is not **"logical-partition"** • No physical processors are operating • The CPC is IMLed in a test or debug mode • An IML is required
	2	The operation was rejected by the Support Element (SE), because the SE is currently performing processing that requires exclusive control of the SE. Retry the operation at a later time.
	297	Some, but not all, of the requested resources were added.
500 (Server Error)	275	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Remove Temporary Capacity

The **Remove Temporary Capacity** operation removes temporary processors or decreases temporary model capacity from the CPC object designated by *{cpc-id}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/remove-temp-capacity

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`record-id`	String (1-8)	Required	Identifies the capacity record to be used for this request
`software-model`	String (1-3)	Optional	The target software model. This value must be one of the software models defined within the capacity record. Implicit in this value is the number of general processors removed. If not provided, the current software model is not changed.
`processor-info`	Array of processor-info objects	Optional	A nested object that defines the number of specialty processors to be removed. If not provided, the number of specialty processors is not changed. Refer to "Request body contents" on page 541 of the **Add Temporary Capacity** operation for details.

Description

When you are finished using all or part of a capacity upgrade, you can remove processors or decrease model capacity using this operation. You can only remove activated resources for the specific offering. You cannot remove dedicated engines or the last processor of a processor type.

If you remove resources back to the base configuration, the capacity record activation is completed. That is, if you remove the last temporary processor, your capacity record is deactivated. For a CBU and On/Off CoD record, to add resources again, you must use another **Add Temporary Capacity** operation. For an On/Off CoD test or CPE record, once the record is deactivated, it is no longer available for use. You can then delete the record.

After removal of the resources, the capacity record remains as an installed record. If you want a record deleted, you must manually select the record on the Installed Records page and click Delete.

Refer to the *Capacity on Demand User's Guide* for details on temporary capacity changes.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the CIM Actions Deactivate task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	271	A duplicate processor-type entry was found in the processor-info array, remove the duplicate entry.
	276	Either the request specifies more resources than are currently active or the requested software model specifies more resources than the current software model.
	278	The software-model value was not found in the capacity record. Only software models as defined in the target capacity record can be specified.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	274	The requested capacity record does not exist.
409 (Conflict)	1	The operation is unavailable in the current CPC state: • The SE is not configured to allow temporary capacity changes via an API • The CPC **status** property is not **"operating"** or the CPC iml-mode property is not **"logical-partition"**.
500 (Server Error)	275	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Swap Current Time Server

The **Swap Current Time Server** operation changes the role of the CPC object designated by *{cpc-id}* to the Current Time Server (CTS).

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/swap-cts

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
stp-id	String (1-8)	Required	Identifies the STP. Can contain 0-9, a-z, A-Z, underline (_) and dash (-).

Description

This operation changes the role of the CPC object designated by *{cpc-id}* to the Current Time Server (CTS).

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the **System (Sysplex) Time** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	285	The CPC targeted by this operation is already the Preferred Time Server.
	286	The operation was rejected for one of the following reasons: • The stp-id field value does not match the current CTN identifier • The operation is not permitted for a mixed CTN.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The requested operation cannot be performed, due to the state of the object: • Server Time Protocol is not enabled on this CPC • an ETR reverse migration is in progress • no alternate is active • this CPC is not the backup time server

HTTP error status code	Reason code	Description
500 (Server Error)	272	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Set STP Configuration

The **Set STP Configuration** operation updates the configuration for an STP-only Coordinated Timing Network.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/set-stp-config

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
stp-id	String (1-8)	Required	The current STP identifier for the CTN, used to verify that the CPC is a member of the correct CTN. Can contain 0-9, a-z, A-Z, underline (_) and dash (-).
new-stp-id	String (1-8)	Optional	If provided, the new STP identifier for the CTN, Can contain 0-9, a-z, A-Z, underline (_) and dash (-).
force	Boolean	Required	Whether a disruptive operation is allowed (true) or rejected (false)
preferred-time-server	stp-node object	Required	Identifies the CPC object to be the Preferred Time Server. Refer to Table 119 on page 489 for details.
backup-time-server	stp-node object	Optional	Identifies the CPC object to be the Backup Time Server. If not provided, the STP has no Backup Time Server. Refer to Table 119 on page 489 for details.
arbiter	stp-node object	Optional	Identifies the CPC object to be the Arbiter for the CTN. If not provided, the STP has no Arbiter. Refer to Table 119 on page 489 for details.
current-time-server	String Enum	Required	Identifies the role of the Current Time Server (CTS). One of: • **"preferred"** - the Preferred Time Server is the CTS • **"backup"** - the Backup Time Server is the CTS.

Description

The CPC object designated by *{cpc-id}* must be the system that becomes the Current Time Server (CTS).

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*

- Action/task permission to the **System (Sysplex) Time** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	282	The operation was rejected, due to an incomplete preferred, backup or arbiter nested object specification. Refer to Table 119 on page 489 for details.
	284	This operation does not target the Current Time Server CPC.
	285	The operation was rejected, due to one of the following configuration errors: • A backup-time-server object is required when providing an arbiter object • A backup-time-server object is required when current-time-server is backup • The preferred-time-server, backup-time-server and arbiter objects do not reference different CPCs
	286	The operation was rejected for one of the following reasons: • The stp-id field value does not match the current CTN identifier • The operation is not permitted for a mixed CTN
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	2	An object URI in one of the stp-node objects in the request body does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The requested operation cannot be performed, due to the state of the object: • Server Time Protocol is not enabled on this CPC • an ETR reverse migration is in progress • no alternate is active • the operation is not permitted for a mixed-CTN.
	287	The provided configuration can only be set by specifying force=true.
	288	No communication path between preferred-time-server and backup-time-server.
	289	No communication path to the arbiter.
500 (Server Error)	272	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Change STP-only Coordinated Timing Network

The **Change STP-only Coordinated Timing Network** operation, sent to the CPC object designated by *{cpc-id}* with the role of Current Time Server (CTS) in an STP-only Coordinated Timing Network (CTN), changes the STP ID portion of the CTN ID for the entire STP-only CTN.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/change-stponly-ctn

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`stp-id`	String (1-8)	Required	The new STP identifier. Can contain 0-9, a-z, A-Z, underline (_) and dash (-).

Description

This operation, sent to the CPC object designated by *{cpc-id}* with the role of Current Time Server (CTS) in an STP-only Coordinated Timing Network (CTN), changes the STP ID portion of the CTN ID for the entire STP-only CTN.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the **System (Sysplex) Time** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	284	The CPC targeted by the operation is not the Current Time Server. Retry the operation using the object-uri for the Current Time Server CPC.
	286	The operation is not permitted for a mixed CTN.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The requested operation cannot be performed, due to the state of the object: • Server Time Protocol is not enabled on this CPC • an ETR reverse migration is in progress
500 (Server Error)	272	An unexpected error occurred during processing of the Server Time Protocol operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Join STP-only Coordinated Timing Network

The **Join STP-only Coordinated Timing Network** operation allows a CPC object designated by *{cpc-id}* to join an STP-only Coordinated Timing Network (CTN).

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/join-stponly-ctn

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
stp-id	String (1-8)	Required	Identifies the STP to be joined. Can contain 0-9, a-z, A-Z, underline (_) and dash (-).

Description

If the CPC object is already participating in a different STP-only CTN and is the Current Time Server (CTS), the operation is rejected. Otherwise, the CPC object is removed from its current CTN and joins the specified CTN.

If the CPC object has an ETR ID, the ETR ID is removed.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the **System (Sysplex) Time** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The requested operation cannot be performed, due to the state of the object: • Server Time Protocol is not enabled on this CPC

HTTP error status code	Reason code	Description
500 (Server Error)	272	An unexpected error occurred during processing of the Server Time Protocol operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Leave STP-only Coordinated Timing Network

The **Leave STP-only Coordinated Timing Network** operation allows a CPC object designated by *{cpc-id}* to leave the STP-only Coordinated Timing Network (CTN) in which it currently participates.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/operations/leave-stponly-ctn

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Description

The CPC object cannot be the Current Time Server (CTS) in the CTN in which it is currently participating.

On success, HTTP status code 204 (No Content) is returned.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission to the **System (Sysplex) Time** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	286	The operation is not permitted for a mixed CTN.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
409 (Conflict)	1	The requested operation cannot be performed, due to the state of the object: • Server Time Protocol is not enabled on this CPC • this CPC is not a member of a CTN • this CPC is the Current Time Server.

HTTP error status code	Reason code	Description
500 (Server Error)	272	An unexpected error occurred during processing of the Server Time Protocol operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Logical partition object

The Processor Resource/Systems Manager™ (PR/SM) is a feature of IBM mainframes that enables logical partitioning of the CEC. A logical partition (LPAR) is a virtual machine at the hardware level. Each LPAR operates as an independent server running its own operating environment. Each LPAR runs its own operating system, which can be any mainframe operating system.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This object includes the properties defined in "Base managed object properties schema" on page 33, with the following class-specific specialization:

Table 139. Logical Partition object: base managed object properties specializations

Name	Qualifier	Type	Description of specialization
object-uri	—	String/ URI	The canonical URI path of the Logical Partition object, of the form `/api/logical-partitions/{logical-partition-id}` where *{logical-partition-id}* is the value of the **object-id** property of the Logical Partition object.
parent	—	String/ URI	The canonical URI path of the associated CPC object.
class	—	String	The class of a Logical Partition object is **"logical-partition"**.
name	(ro)	String (1-8)	The name of the logical partition
description	(ro)	String (0-1024)	The descriptive text associated with this object.
status	(sc)	String Enum	One of the following values: • **"operating"** - the logical partition has a active control program • **"not-operating"** - the logical partition's CPC is non operational • **"not-activated"** - the logical partition does not have an active control program • **"exceptions"** - the logical partition's CPC has one or more unusual conditions
acceptable-status	(w)(pc)	Array of String Enum	An array of one or more status strings that determine an acceptable status for a logical partition. When a logical partition's **status** property contains one of the specified acceptable-status values, the **has-unacceptable-status** property contains false.

Class specific additional properties

In addition to the properties defined via included schemas, this object includes the following additional class-specific properties. Refer to the *Processor Resource/Systems Manager Planning Guide* for more detailed explanations of the various properties.

There are additional notes throughout the table. Please refer to the note list at the end of the table.

Table 140. Logical Partition object: class specific additional properties

Name	Qualifier	Type	Description
os-name[1]	(pc)	String (0-8)	An operating system provided value, used to identify the operating system instance. The format of the value is operating system dependant. If not provided by the operating system, an empty string is returned.
os-type[1]	(pc)	String (0-8)	A human readable form of the operating system provided value for the type of the operating system active in this logical partition. If not provided, an empty string is returned.
os-level[1]	(pc)	String (0-32)	A human readable form of the operating system provided value for the level of the operating system active in this logical partition. If not provided, an empty string is returned.
sysplex-name[1]	(pc)	String (1-8)	Applicable only for z/OS, the name of the sysplex to which this logical partition is a member. Otherwise a null object is returned.
has-operating-system-messages[1]	—	Boolean	If true, object has operating system messages. If false, object does not have operating system messages.
activation-mode[1]	—	String Enum	One of the following values: • **"esa390"** - the logical partition is in ESA/390 mode • **"esa390tpf"** - the logical partition is in ESA/390 TPF mode • **"coupling-facility"** - the logical partition is running as a coupling facility • **"linux"** - the logical partition is in Linux mode • **"zvm"** - the logical partition is in z/VM mode • **"zaware"** - the logical partition is in IBM zAware mode.
next-activation-profile-name	(w)(pc)	String (1-16)	Image activation profile name or load activation profile name to be used on the next activate.
last-used-activation-profile	(pc)	String (0-16)	The last used activation profile name or a null string.
initial-processing-weight[1, 2, 3]	(w)	Integer	The relative amount of shared general purpose processor resources allocated to the logical partition: Get: **0** The **object-id** does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the relative amount of shared general purpose processor resources allocated to the logical partition. Update: **1-999** Defines the relative amount of shared general purpose processor resources allocated to the logical partition.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
initial-processing-weight-capped[1, 2, 3, 4]	(w)	Boolean	Whether the initial processing weight for general purpose processors is a limit or a target. **True** Indicates that the initial general purpose processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of general purpose processor resources. **False** Indicates that the initial general purpose processor processing weight for the logical partition is not capped. It represents the share of general purpose processor resources guaranteed to a logical partition when all general purpose processor resources are in use. Otherwise, when excess general purpose processor resources are available, the logical partition can use them if necessary.
minimum-processing-weight[1, 2, 3]	(w)	Integer	The minimum relative amount of shared general purpose processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the minimum relative amount of shared general purpose processor resources allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared general purpose processor resources allocated to the logical partition. The value must be less than or equal to the initial-processing-weight property.
maximum-processing-weight[1, 2, 3]	(w)	Integer	The maximum relative amount of shared general purpose processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the maximum relative amount of shared general purpose processor resources allocated to the logical partition. Update: **1-999** Defines the maximum relative amount of shared general purpose processor resources allocated to the logical partition. The value must be greater than or equal to the initial-processing-weight property.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
current-processing-weight[1, 3]	—	Integer	The relative amount of shared general purpose processor resources currently allocated to the logical partition. **0** The **object-id** does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the relative amount of shared general purpose processor resources currently allocated to the logical partition.
current-processing-weight-capped[1, 2, 3]	—	Boolean	Whether the current general purpose processing weight is a limit or a target. **True** Indicates that the current general purpose processing weight for the logical partition is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources. **False** Indicates that the current general purpose processing weight for the logical partition is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.
workload-manager-enabled[1, 5]	(w)	Boolean	Whether or not z/OS Workload Manager is allowed to change processing weight related properties. **True** Indicates that z/OS Workload Manager is allowed to change processing weight related properties for this logical partition. **False** Indicates that z/OS Workload Manager is not allowed to change processing weight related properties for this logical partition.
defined-capacity[1]	(w)	Integer	The defined capacity expressed in terms of Millions of Service Units (MSU)s per hour. MSU is a measure of processor resource consumption. The amount of MSUs a logical partition consumes is dependent on the model, the number of logical processors available to the partition, and the amount of time the logical partition is dispatched. The defined capacity value specifies how much capacity the logical partition is to be managed to by z/OS Workload Manager for the purpose of software pricing. **0** No defined capacity is specified for this logical partition. **1-nnnn** Represents the amount of defined capacity specified for this logical partition.
cluster-name[1]	(pc)	String (0-8)	LPAR cluster name, which identifies membership in a group of logical partitions that are members of the same z/OS Parallel Sysplex®.
partition-number[1]	(pc)	String (2)	The partition number for the logical partition, in hexadecimal.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
initial-aap-processing-weight[1, 3, 6]	(w)	Integer	The relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Update: **1-999** Define the relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.
initial-aap-processing-weight-capped[1, 3, 4, 6, 7]	(w)	Boolean	Whether the initial processing weight for Application Assist Processor (zAAP) processors is a limit or a target. **True** Indicates that the initial Application Assist Processor (zAAP) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Application Assist Processor (zAAP) processor resources, regardless of the availability of excess Application Assist Processor (zAAP) processor resources. **False** Indicates that the initial Application Assist Processor (zAAP) processor processing weight for the logical partition is not capped. It represents the share of Application Assist Processor (zAAP) processor resources guaranteed to a logical partition when all Application Assist Processor (zAAP) processor resources are in use. Otherwise, when excess Application Assist Processor (zAAP) processor resources are available, the logical partition can use them if necessary.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
minimum-aap-processing-weight[1, 3, 6]	(w)	Integer	The minimum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the minimum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Update: **0** No minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.
maximum-aap-processing-weight[1, 3, 6]	(w)	Integer	The maximum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the maximum relative amount of shared Application Assist Processor (zAAP) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.
current-aap-processing-weight[1, 3]	—	Integer	The current relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. **0** The **object-id** does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the current relative amount of shared Application Assist Processor (zAAP) processor resources initially allocated to the logical partition.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
current-aap-processing-weight-capped[1, 3, 6, 7]	—	Boolean	Whether the current Application Assist Processor (zAAP) processing weight is a limit or a target. **True** Indicates that the current Application Assist Processor (zAAP) processing weight for the logical partition is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources. **False** Indicates that the current Application Assist Processor (zAAP) processing weight for the logical partition is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.
initial-ifl-processing-weight[1, 3, 8]	(w)	Integer	The relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Update: **1-999** Define the relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.
initial-ifl-processing-weight-capped[1, 3, 4, 8, 9]	(w)	Boolean	Whether the initial processing weight for Integrated Facility for Linux (IFL) processors is a limit or a target. **True** Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Integrated Facility for Linux (IFL) processor resources, regardless of the availability of excess Integrated Facility for Linux (IFL) processor resources **False** Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the logical partition is not capped. It represents the share of Integrated Facility for Linux (IFL) processor resources guaranteed to a logical partition when all Integrated Facility for Linux (IFL) processor resources are in use. Otherwise, when excess Integrated Facility for Linux (IFL) processor resources are available, the logical partition can use them if necessary.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
minimum-ifl-processing-weight[1, 3, 8]	(w)	Integer	The minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.
maximum-ifl-processing-weight[1, 3, 8]	(w)	Integer	The maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.
current-ifl-processing-weight[1, 3]	—	Integer	The current relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. **0** The **object-id** does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the current relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the logical partition.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
current-ifl-processing-weight-capped[1, 3, 8, 9]	—	Boolean	Whether the current Integrated Facility for Linux (IFL) processing weight is a limit or a target. **True** Indicates that the current Integrated Facility for Linux (IFL) processing weight for the logical partition is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources. **False** Indicates that the current Integrated Facility for Linux (IFL) processing weight for the logical partition is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.
initial-ziip-processing-weight[1, 3, 10]	(w)	Integer	The relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Update: **1-999** Define the relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
initial-ziip-processing-weight-capped[1, 3, 4, 10, 11]	(w)	Boolean	Whether the initial processing weight for Integrated Information Processors (zIIP) processors is a limit or a target. **True** Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Integrated Information Processors (zIIP) processor resources, regardless of the availability of excess Integrated Information Processors (zIIP) processor resources. **False** Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the logical partition is not capped. It represents the share of Integrated Information Processors (zIIP) processor resources guaranteed to a logical partition when all Integrated Information Processors (zIIP) processor resources are in use. Otherwise, when excess Integrated Information Processors (zIIP) processor resources are available, the logical partition can use them if necessary.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
minimum-ziip-processing-weight[1, 3, 10]	(w)	Integer	The minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
maximum-ziip-processing-weight[1, 3, 10]	(w)	Integer	The maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Get: **0** The **object-id** does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
current-ziip-processing-weight[1, 3]	—	Integer	The current relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. **0** The **object-id** does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the current relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the logical partition.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
current-ziip-processing-weight-capped[1, 3, 10, 11]	—	Boolean	Whether the current Integrated Information Processors (zIIP) processing weight is a limit or a target. **True** Indicates that the current Integrated Information Processors (zIIP) processing weight for the logical partition is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources. **False** Indicates that the current Integrated Information Processors (zIIP) processing weight for the logical partition is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.
initial-cf-processing-weight[1, 3, 12]	(w)	Integer	The relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Get: **0** The **object-id** does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Update: **1-999** The relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object.
initial-cf-processing-weight-capped[1, 3, 4, 12, 13]	(w)	Boolean	Indicates whether the initial processing weight for Internal Coupling Facility (ICF) processors is a limit or a target. **True** Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the Logical Partition object is capped. It represents the logical partition's maximum share of Internal Coupling Facility (ICF) processor resources, regardless of the availability of excess Internal Coupling Facility (ICF) processor resources. **False** Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the Logical Partition is not capped. It represents the share of Internal Coupling Facility (ICF) processor resources guaranteed to a logical partition when all Internal Coupling Facility (ICF) processor resources are in use. Otherwise, when excess Internal Coupling Facility (ICF) processor resources are available, the logical partition can use them if necessary.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
minimum-cf-processing-weight[1, 3, 12]	(w)	Integer	The minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Get: **0** The **object-id** does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Update: **1-999** The minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object.
maximum-cf-processing-weight[1, 3, 12]	(w)	Integer	The maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Get: **0** The **object-id** does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. Update: **1-999** Define the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object.
current-cf-processing-weight[1, 3]	—	Integer	The current relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object. **0** The **object-id** does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the current relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the Logical Partition object.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
current-cf-processing-weight-capped[1, 3, 12, 13]	—	Boolean	Indicates whether the current Internal Coupling Facility (ICF) processing weight is a limit or a target. **True** Indicates that the current Internal Coupling Facility (ICF) processing weight for the Logical Partition object is capped. It represents the logical partition's maximum share of resources, regardless of the availability of excess processor resources. **False** Indicates that the current Internal Coupling Facility (ICF) processing weight for the Logical Partition is not capped. It represents the share of resources guaranteed to a logical partition when all processor resources are in use. Otherwise, when excess processor resources are available, the logical partition can use them if necessary.
program-status-word-information[1]	—	Array of psw-description objects	Describes the current PSW information for each CP associated with the logical partition. The information is obtained on each Get Logical Partition Properties request and is not cached. Refer to the description of the psw-description object for details.
os-ipl-token[1]	(pc)	String (1-16)	Applicable only to z/OS, a value provided when z/OS is IPLed that uniquely identifies the instance of the operating system. Used by z/OS to obtain knowledge about the status of another system in the sysplex, and upon the demise of the system, potentially partition the system out of the sysplex immediately and reset the demised system. The value is a string of hexadecimal characters (0-9,A-Z), left justified.
group-profile-capacity[1]	(w)	Integer	The current capacity value of the Group Profile with which the logical partition is associated. A null object is returned if the logical partition is not assigned to an LPAR group.
group-profile-uri[1]	—	String/URI	The canonical URI of the Group Profile associated with the logical partition. A null object is returned if the logical partition is not assigned to an LPAR group.
zaware-host-name[14]	(w)	String (1-64)	The IBM zAware host name. Valid characters are: a-z,A-Z,0-9, period(.), minus(-) and colon(:)
zaware-master-userid[14]	(w)	String (1-32)	The IBM zAware master userid. Valid characters are: a-z,A-Z,0-9, period(.), minus(-) and underscore (_)
zaware-master-pw[14]	(w)	String 98-256)	The IBM zAware master password. Valid characters are: a-z,A-Z,0-9 and !@#$%^&*()_+{}|<>?-= This property is not returned on a Get request, it can only be specified on an Update request.
zaware-network-info[14]	(w)	Array of zaware-network objects	The set of networks available to IBM zAware. A minimum of 1 network and a maximum of 100 networks are permitted. On an Update request, this property fully replaces the existing set.
zaware-gateway-info[14]	(w)	ip-info object	The default gateway IP address information.

Table 140. Logical Partition object: class specific additional properties (continued)

Name	Qualifier	Type	Description
zaware-dns-info[14]	(w)	Array of ip-info objects	The DNS IP address information. A minimum of 0 entries and a maximum of 2 entries are permitted. On an Update request, this property fully replaces the existing set.

Notes:

1. If the logical partition **status** property is **"not-activated"**, a null object is returned instead of the documented field type.
2. An Update of this property is only valid for an **object-id** that represents a logical partition with at least one not dedicated general purpose processor.
3. The value returned from a Get request is a null object for an **object-id** that does not represent a logical partition with at least one not dedicated general purpose processor.
4. This property and the **workload-manager-enabled** property are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this property it might be necessary to first disable the **workload-manager-enabled** property.
5. This property and the various capping properties are mutually exclusive and cannot be enabled at the same time. Therefore in order to enable this property it may be necessary to first disable any capping property that is currently enabled.
6. An Update of this property is only valid for an **object-id** that represents a logical partition with at least one not dedicated Application Assist Processor (zAAP) processor.
7. The value returned from a Get request is always false for an **object-id** that does not represent a logical partition with at least one not dedicated Application Assist Processor (zAAP) processor.
8. An Update of this property is only valid for an **object-id** that represents a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
9. The value returned from a Get request is always false for an **object-id** that does not represent a logical partition with at least one not dedicated Integrated Facility for Linux (IFL) processor.
10. An Update of this property is only valid for an **object-id** that represents a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.
11. The value returned from a Get request is always false for an **object-id** that does not represent a logical partition with at least one not dedicated Integrated Information Processors (zIIP) processor.
12. An Update of this property is only valid for an **object-id** that represents a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
13. The value returned for a Get request is always false when the **object-id** does not represent a logical partition with at least one not dedicated Internal Coupling Facility (ICF) processor.
14. On a Get request, this property is returned only when **activation-mode** is **"zaware"**. On an Update request, this property can be updated only when **activation-mode** is **"zaware"**.

List Logical Partitions of CPC

The **List Logical Partitions of CPC** operation lists the logical partitions of a CPC.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/logical-partitions

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC.

Query Parameters

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`logical-partition`	Array of logical-partition-info objects	Array of nested logical-partition-info objects (described in the next table)

Each nested logical-partition-info object contains the following fields:

Field name	Type	Description
`object-uri`	String/URI	Canonical URI path of the Logical Partition object
`name`	String	The name of the Logical Partition object
`status`	String Enum	The current status of the Logical Partition object

Description

This operation lists the logical partition objects that belong to a CPC. The object URI, object ID and display name are provided for each.

If the **name** query parameter is specified, the returned list is limited to those logical partition objects that have a name property matching the specified filter pattern. If the name parameter is omitted, this filtering is not done.

An object is only included in the list if the API user has object-access permission for that object.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Object access permission to any Logical Partition object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/logical-partitions HTTP/1.1
x-api-session: 65aw2jahugn1wop51hsq0c6aldkkx773dz9ulirrvg2z853m4u
```

Figure 271. List Logical Partitions of CPC: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 16:58:36 GMT
content-type: application/json;charset=UTF-8
content-length: 374
{
   "logical-partitions": [
      {
         "name": "APIVM1",
         "object-uri": "/api/logical-partitions/c7eb8134-826e-3a71-8d1a-00d706c874e9",
         "status": "operating"
      },
      {
         "name": "ZOS",
         "object-uri": "/api/logical-partitions/458e44e1-b0c2-391b-83ff-ecfd847295bd",
         "status": "not-operating"
      }
   ]
}
```

Figure 272. List Logical Partitions of CPC: Response

Get Logical Partition Properties

The **Get Logical Partition Properties** operation retrieves the properties of a single Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

GET /api/logical-partitions/{*logical-partition-id*}

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Response body contents

On successful completion, HTTP status code 200 (OK) is returned and the response body provides the current values of the properties for the Logical Partition object as defined in "Data model" on page 551.

Description

The URI path must designate an existing Logical Partition object and the API user must have object-access permission to it. If either of these conditions is not met, status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined in "Data model" on page 551.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 566.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/logical-partitions/c7eb8134-826e-3a71-8d1a-00d706c874e9 HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 273. Get Logical Partition Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:16 GMT
content-type: application/json;charset=UTF-8
content-length: 2366
{
   "acceptable-status": [
      "operating"
   ],
   "activation-mode": "esa390",
   "additional-status": "",
   "class": "logical-partition",
   "cluster-name": "",
   "current-aap-processing-weight": null,
   "current-aap-processing-weight-capped": null,
   "current-cf-processing-weight": null,
   "current-cf-processing-weight-capped": null,
   "current-ifl-processing-weight": null,
   "current-ifl-processing-weight-capped": null,
   "current-processing-weight": 100,
   "current-processing-weight-capped": false,
   "current-ziip-processing-weight": null,
   "current-ziip-processing-weight-capped": null,
   "defined-capacity": 0,
   "description": "LPAR Image",
   "group-profile-capacity": null,
   "group-profile-uri": null,
   "has-operating-system-messages": false,
   "has-unacceptable-status": false,
   "initial-aap-processing-weight": null,
   "initial-aap-processing-weight-capped": null,
   "initial-cf-processing-weight": null,
   "initial-cf-processing-weight-capped": null,
   "initial-ifl-processing-weight": null,
   "initial-ifl-processing-weight-capped": null,
   "initial-processing-weight": 100,
   "initial-processing-weight-capped": false,
   "initial-ziip-processing-weight": null,
   "initial-ziip-processing-weight-capped": null,
   "is-locked": false,
   "last-used-activation-profile": "APIVM1",
   "maximum-aap-processing-weight": null,
   "maximum-cf-processing-weight": null,
   "maximum-ifl-processing-weight": null,
   "maximum-processing-weight": 200,
   "maximum-ziip-processing-weight": null,
```

Figure 274. Get Logical Partition Properties: Response (Part 1)

```
   "minimum-aap-processing-weight": null,
   "minimum-cf-processing-weight": null,
   "minimum-ifl-processing-weight": null,
   "minimum-processing-weight": 50,
   "minimum-ziip-processing-weight": null,
   "name": "APIVM1",
   "next-activation-profile-name": "APIVM1",
   "object-id": "c7eb8134-826e-3a71-8d1a-00d706c874e9",
   "object-uri": "/api/logical-partitions/c7eb8134-826e-3a71-8d1a-00d706c874e9",
   "os-ipl-token": "0000000000000000",
   "os-level": "6.2.0",
   "os-name": "APIVM1",
   "os-type": "z/VM",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "partition-number": 1,
   "program-status-word-information": [
      {
         "cpid": "00",
         "psw": "0706400080000000000000000000000000"
      },
      {
         "cpid": "01",
         "psw": "0706400080000000000000000000000000"
      }
   ],
   "status": "operating",
   "sysplex-name": "SSICAPI1",
   "workload-manager-enabled": true
}
```

Figure 275. Get Logical Partition Properties: Response (Part 2)

Update Logical Partition Properties

The **Update Logical Partition Properties** operation updates one or more writeable properties of the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Response body contents

The request body is expected to contain one or more field names representing writable logical partition properties, along with the new values for those fields.

The response body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the Logical Partition object type to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the object is updated with the value provided by the input field, and status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Change Object Definition** task.
- Object access permission to the logical partition's parent CPC object.
- For a logical partition whose **activation-mode** is **"zaware"**, action/task permission for the **Firmware Details** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
403 (Forbidden)	1	The API user does not have the required permission for this operation.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Activate Logical Partition

The **Activate Logical Partition** operation activates the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/activate

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`activation-profile-name`	String (1-16)	Optional	The name of the activation profile to be used for the request. If not provided, the request uses the profile name specified in the **next-activation-profile-name** property for the Logical Partition object.

Field name	Type	Rqd/Opt	Description
force	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

Activation is a process that makes a logical partition operational, which means either:
- The logical partition is ready to have a control program or operating system loaded, or
- The logical partition has loaded and is running a control program or operating system.

Activating a logical partition includes:
- Initializing the logical partition
- Allocating system resources to the logical partition
- Loading the logical partition with a control program or operating system.

Since the status of the logical partition determines which operations must be performed during activation to make the logical partition operational, one or more operations listed above may not be performed during activation.

If planning to load the z/VM operating system in this logical partition, refer to Chapter 10, "Virtualization management," on page 161 for details on the activation of virtual servers.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes" on page 572.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Activate** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 571.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current logical partition status and use of the force=false parameter. If rejected due to force=false, the logical partition status is unchanged. If the operation failed, the logical partition status is unknown. Refer to the message parameter in the error response body for details.

Deactivate Logical Partition

The **Deactivate Logical Partition** operation deactivates the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

```
POST /api/logical-partitions/{logical-partition-id}/operations/deactivate
```

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

Deactivation is an orderly process for terminating a logical partition.

Deactivating a logical partition includes:
- Unloading the logical partition's control program or operating system
- Freeing system resources allocated to the logical partition.

After the logical partition is deactivated, the logical partition is no longer operational

If this logical partition currently has the z/VM operating system loaded, refer to Chapter 10, "Virtualization management," on page 161 for the details on deactivation of virtual servers.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes" on page 574.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Deactivate** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current logical partition status and use of the force=false parameter. If rejected due to force=false, the logical partition status is unchanged. If the operation failed, the logical partition status is unknown. Refer to the message parameter in the error response body for details.

Reset Normal

The **Reset Normal** operation initializes a system or logical partition by clearing its pending interruptions, resetting its channel subsystem and resetting its processors. A reset prepares a system or logical partition for loading it with an operating system.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/reset-normal

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.
`os-ipl-token`	String (1-16)	Optional	Applicable only to z/OS, this parameter requests that this operation only be performed if the provided value matches the current value of the **os-ipl-token** property. This ensures that this operation is targeting the same IPL instance as when the **os-ipl-token** property was retrieved. IBM recommends that this parameter only be provided by callers that fully understand how the **os-ipl-token** parameter is managed by z/OS. The value is a string of hexadecimal characters (0-9, A-Z), left justified.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes" on page 576.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Reset Normal** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	264	The specified IPL Token value does not match the current IPL Token value.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current logical partition status and use of the force=false parameter. If rejected due to force=false, the logical partition status is unchanged. If the operation failed, the logical partition status is unknown. Refer to the message parameter in the error response body for details.

Reset Clear

The **Reset Clear** operation initializes system or logical partition by clearing its pending interruptions, resetting its channel subsystem and resetting its processors. A reset prepares a system or logical partition for loading it with an operating system and clears main memory of the system or logical partition.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/reset-clear

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`force`	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.
`os-ipl-token`	String (1-16)	Optional	Applicable only to z/OS, this parameter requests that this operation only be performed if the provided value matches the current value of the **os-ipl-token** property. This ensures that this operation is targeting the same IPL instance as when the **os-ipl-token** property was retrieved. IBM recommends that this parameter only be provided by callers that fully understand how the **os-ipl-token** parameter is managed by z/OS. The value is a string of hexadecimal characters (0-9,A-Z), left justified.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
`job-uri`	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes."

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Reset Clear** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 576.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	264	The specified IPL Token value does not match the current IPL Token value.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current logical partition status and use of the force=false parameter. If rejected due to force=false, the logical partition status is unchanged. If the operation failed, the logical partition status is unknown. Refer to the message parameter in the error response body for details.

Load Logical Partition

The **Load Logical Partition** operation resets a logical partition, to prepare it for loading an operating system, and loads the operating system.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/load

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
load-address	String (1-5)	Required	The hexadecimal address of an I/O device that provides access to the control program to be loaded. The input value is right justified and padded with zeros to 5 characters. Valid values are in the range "00000" to "nFFFF" where "n" is the number of subchannel sets provided by the CPC minus 1. So, for example, on a CPC that provides 3 subchannel sets, the valid range is "00000" to "2FFFF".
load-parameter	String (1-8)	Optional	Some control programs support the use of this property to provide additional control over the outcome of a **Load** operation. Refer to the configuration documentation for the control program to be loaded to see if this parameter is supported and if so, what values and format is supported. Omitting this field indicates that the value for this field is to be retrieved from the current IOCDS. Valid characters are 0-9, A-Z, blank and period.
clear-indicator	Boolean	Optional	Whether memory should be cleared before performing the **Load** (true) or not cleared (false). The default value is true.
timeout	Integer (60-600)	Optional	Amount of time, in seconds, to wait for the **Load** to complete. The default timeout value is 60 seconds.
store-status-indicator	Boolean	Optional	Whether status should be stored before performing the **Load** (true) or not stored (false). The default is false.
force	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.
os-ipl-token	String (1-16)	Optional	Applicable only to z/OS, this parameter requests that this operation only be performed if the provided value matches the current value of the **os-ipl-token** property. This ensures that this operation is targeting the same IPL instance as when the **os-ipl-token** property was retrieved. IBM recommends that this parameter only be provided by callers that fully understand how the **os-ipl-token** parameter is managed by z/OS. The value is a string of hexadecimal characters (0-9, A-Z), left justified.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

This operation is not permitted for a logical partition whose **activation-mode** property is **"zaware"**.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes" on page 580.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Load** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 578.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	264	The specified IPL Token value does not match the current IPL Token value.
	306	This operation is not valid in the current activation mode.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed or was rejected due to the current logical partition status and use of the force=false parameter. If rejected due to force=false, the logical partition status is unchanged. If the operation failed, the logical partition status is unknown. Refer to the message parameter in the error response body for details.

PSW Restart

The **PSW Restart** operation restarts the first available processor of the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/psw-restart

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

This operation is not permitted for a logical partition whose **activation-mode** property is **"zaware"**.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See “Query Job Status” on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See “Job status and reason codes” on page 581.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **PSW Restart** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 580.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	306	This operation is not valid in the current activation mode.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed.

Start Logical Partition

The **Start Logical Partition** operation starts the processors to process instructions of the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/start

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

This operation is not permitted for a logical partition whose **activation-mode** property is **"zaware"**.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See "Query Job Status" on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See "Job status and reason codes."

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Start** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 581.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	306	This operation is not valid in the current activation mode.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed.

Stop Logical Partition

The **Stop Logical Partition** operation stops the processors from processing instructions of the Logical Partition object designated by *{logical-partition-id}*.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/stop

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

This operation is not permitted for a logical partition whose **activation-mode** property is **"zaware"**.

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See “Query Job Status” on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See “Job status and reason codes” on page 584.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **Stop** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in “Response body contents.”

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	306	This operation is not valid in the current activation mode.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed.

SCSI Load

The **SCSI Load** operation clears main storage, to prepare the logical partition for loading an operating system, and loads the operating system from the designated SCSI device.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/scsi-load

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`load-address`	String (1-5)	Required	The hexadecimal address of an I/O device that provides access to the control program to be loaded. The input value is right justified and padded with zeros to 5 characters. Valid values are in the range "00000" to "nFFFF" where "n" is the number of subchannel sets provided by the CPC minus 1. So, for example, on a CPC that provides 3 subchannel sets, the valid range is "00000" to "2FFFF".
`load-parameter`	String (1-8)	Optional	Some control programs support the use of this property to provide additional control over the outcome of a **Load** operation. Refer to the configuration documentation for the control program to be loaded to see if this parameter is supported and if so, what values and format is supported. Omitting this field indicates that the value for this field is to be retrieved from the current IOCDS. Valid characters are 0-9, A-Z, blank and period

Field name	Type	Rqd/Opt	Description
world-wide-port-name	String (1-16)	Required	The worldwide port name (WWPN) of the target SCSI device to be used for this operation, in hexadecimal.
logical-unit-number	String (1-16)	Required	The hexadecimal logical unit number (LUN) to be used for the **SCSI Load**.
disk-partition-id	Integer (0-30)	Optional	The disk-partition-id (also called the boot program selector) to be used for the **SCSI Load**. The default value is 0.
operating-system-specific-load-parameters	String (1-256)	Optional	The operating system specific load parameters to be used for the **SCSI Load**. The default value is an empty string.
boot-record-logical-block-address	String (1-16)	Optional	The hexadecimal boot record logical block address to be used for the **SCSI Load**. The default value is hex zeros.
force	Boolean	Optional	Whether this operation is permitted when the logical partition is in **"operating"** status (true) or not (false). The default is false.
os-ipl-token	String (1-16)	Optional	Applicable only to z/OS, this parameter requests that this operation only be performed if the provided value matches the current value of the **os-ipl-token** property. This ensures that this operation is targeting the same IPL instance as when the **os-ipl-token** property was retrieved. IBM recommends that this parameter only be provided by callers that fully understand how the **os-ipl-token** parameter is managed by z/OS. The value is a string of hexadecimal characters (0-9, A-Z), left justified.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See “Query Job Status” on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See “Job status and reason codes” on page 586.

Authorization requirements

This operation has the following authorization requirements:

- Object access permission to the Logical Partition object designated by *{logical-partition-id}*

- Action/task permission for the **Load** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 585.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed.

SCSI Dump

The **SCSI Dump** operation loads a standalone dump program from a designated SCSI device.

HTTP method and URI

POST /api/logical-partitions/{*logical-partition-id*}/operations/scsi-dump

In this request, the URI variable *{logical-partition-id}* is the object ID of the target Logical Partition object.

Request body contents

The request body is expected to contain a JSON object with the following fields:

Field name	Type	Rqd/Opt	Description
`load-address`	String (1-5)	Required	The hexadecimal address of an I/O device that provides access to the control program to be loaded. The input value is right justified and padded with zeros to 5 characters. Valid values are in the range "00000" to "nFFFF" where "n" is the number of subchannel sets provided by the CPC minus 1. So, for example, on a CPC that provides 3 subchannel sets, the valid range is "00000" to "2FFFF".

Field name	Type	Rqd/Opt	Description
load-parameter	String (1-8)	Optional	Some control programs support the use of this property to provide additional control over the outcome of a **Load** operation. Refer to the configuration documentation for the control program to be loaded to see if this parameter is supported and if so, what values and format is supported. Omitting this field indicates that the value for this field is to be retrieved from the current IOCDS. Valid characters are 0-9, A-Z, blank and period.
world-wide-port-name	String (1-16)	Required	The worldwide port name (WWPN) of the target SCSI device to be used for this operation, in hexadecimal.
logical-unit-number	String (1-16)	Required	The hexadecimal logical unit number (LUN) to be used for the **SCSI Load**.
disk-partition-id	Integer (0-30)	Optional	The disk-partition-id (also called the boot program selector) to be used for the **SCSI Load**. The default value is 0.
operating-system-specific-load-parameters	String (1-256)	Optional	The operating system specific load parameters to be used for the **SCSI Load**. The default value is an empty string.
boot-record-logical-block-address	String (1-16)	Optional	The hexadecimal boot record logical block address to be used for the **SCSI Load**. The default value is hex zeros.
os-ipl-token	String (1-16)	Optional	Applicable only to z/OS, this parameter requests that this operation only be performed if the provided value matches the current value of the **os-ipl-token** property. This ensures that this operation is targeting the same IPL instance as when the **os-ipl-token** property was retrieved. IBM recommends that this parameter only be provided by callers that fully understand how the **os-ipl-token** parameter is managed by z/OS. The value is a string of hexadecimal characters (0-9, A-Z), left justified.

Response body contents

Once the operation is accepted, the response body contains a JSON object with the following fields:

Field name	Type	Description
job-uri	String/URI	URI that may be queried to retrieve status updates.

Asynchronous result description

Once the operation has completed, a job-completion notification is sent and results are available for the asynchronous portion of this operation. These results are retrieved using the **Query Job Status** operation directed at the job URI provided in the response body.

The result document returned by the **Query Job Status** operation is specified in the description for the **Query Job Status** operation. When the status of the job is **"complete"**, the results include a job completion status code and reason code (fields **job-status-code** and **job-reason-code**) which are set as indicated in operation description. The **job-results** field is null for this operation.

Description

When the operation is initiated, a 202 (Accepted) status code is returned. The response body includes a URI that may be queried to retrieve the status of the operation. See “Query Job Status” on page 44 for information on how to query job status. When the operation has completed, an asynchronous result message is sent. See “Job status and reason codes” on page 588.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the Logical Partition object designated by *{logical-partition-id}*
- Action/task permission for the **SCSI Dump** task.
- Object access permission to the logical partition's parent CPC object.

HTTP status and reason codes

On success, HTTP status code 202 (Accepted) is returned and the response body is provided as described in "Response body contents" on page 587.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{logical-partition-id}*) does not designate an existing Logical Partition object, or the API user does not have object access permission to the object.
500 (Server Error)	280	An IO exception occurred during the scheduling of the asynchronous request.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Job status and reason codes

HTTP error status code	Reason code	Description
204 (No Content)	N/A	Operation completed successfully.
500 (Server Error)	263	Operation failed.

Reset activation profile

A Reset activation profile is used by a CPC Activate operation to control the activation of a CPC and, if properly configured with one or more image activation profiles, a set of Logical Partition(s).

An activation profile can only be created or deleted from the Hardware Management Console or the Support Element.

Refer to the *Support Element Operations Guide* for details on customizing activation profiles.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This element includes the following type-specific properties.

Table 141. Reset activation profile: type-specific properties

Name	Qualifier	Type	Description of specialization
element-uri	—	String/ URI	The canonical URI path of the Reset Activation Profile object, of the form `/api/cpcs/{cpc-id}/reset-activation-profiles/{reset-activation-profile-name}` where *{reset-activation-profile-name}* is the value of the name property (Reset Activation Profile name).
parent	—	String/ URI	The canonical URI path of the associated CPC object.
class	—	String	The class of a Reset Activation Profile object is **"reset-activation-profile"**.
name	—	String (1-16)	The activation profile name, which uniquely identifies this profile within the set of activation profiles for the CPC object designated by *{cpc-id}*.
description	(w)	String (1-50)	The reset profile description
iocds-name	(w)	String (0-2)	The Input/Output Configuration Data Set name, in hexadecimal. An empty string indicates that the currently active IOCDS will be used. The active IOCDS is the one from the most recent power-on-reset of the CPC or, if using dynamic I/O configuration, the one last activated.
processor-running-time-type	(w)	String Enum	Defines whether the processor running time is determined dynamically or set manually for the CPC (see processor-running-time in this table). One of: • **"system-determined"** • **"user-determined"**
processor-running-time	(w)	Integer (0-100)	Amount of continuous time, in milliseconds, for logical processors to perform jobs on shared processors for the CPC, if **processor-running-time-type** is set to **"user-determined"**. If **processor-running-time-type** is **"system-determined"**, this property's value will always be returned as 0.
end-timeslice-on-wait	(w)	Boolean	If true and if **processor-running-time-type** is set to **"user-determined"**, CPC Logical Partitions lose their share of running time when they enter a wait state. If **processor-running-time-type** is **"system-determined"**, this property's value will always be returned as false.

List Reset Activation Profiles

The **List Reset Activation Profiles** operation lists the Reset Activation Profiles associated with a particular CPC.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/reset-activation-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`reset-activation-profiles`	Array of reset-actprof-info objects	Array of nested objects (described in the following table).

Each reset-actprof-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the Reset Activation Profile object.
`name`	String	The name of the Reset Activation Profile.

Description

This operation lists the Reset Activation Profiles associated with a particular CPC.

If the **name** query parameter is specified, the returned list is limited to those Reset Activation Profiles that have a name property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	281	An unexpected error occurred during the collection of the list of activation profiles.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/reset-activation-profiles HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 276. List Reset Activation Profiles: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:16 GMT
content-type: application/json;charset=UTF-8
content-length: 372
{
   "reset-activation-profiles": [
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/reset-activation-profiles/
       DEFAULT",
         "name": "DEFAULT"
      },
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/reset-activation-profiles/
        POWER_ON_RESET",
         "name": "POWER_ON_RESET"
      }
   ]
}
```

Figure 277. List Reset Activation Profiles: Response

Get Reset Activation Profile Properties

The **Get Reset Activation Profile Properties** operation retrieves the properties of a single Reset Activation Profile designated by *{reset-activation-profile-name}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/reset-activation-profiles/{*reset-activation-profile-name*}

URI variables:

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{reset-activation-profile-name}	Reset Activation Profile name

Response body contents

On successful completion, the response body provides the current values of the properties for the Reset Activation Profile as defined in the "Data model" on page 588.

Description

The URI path must designate an existing Reset Activation Profile and the API user must have object-access permission to the CPC. If either of these conditions is not met, status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined by the Data Model for the Reset Activation Profile object.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 591.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{reset-activation-profile-name}*) does not designate an existing activation profile.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/reset-activation-profiles/DEFAULT HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 278. Get Reset Activation Profile Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:18 GMT
content-type: application/json;charset=UTF-8
content-length: 384
{
   "class": "reset-activation-profile",
   "description": "This is the default Reset profile.",
   "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/reset-activation-profiles/
    DEFAULT",
   "end-timeslice-on-wait": false,
   "iocds-name": "a0",
   "name": "DEFAULT",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "processor-running-time": 0,
   "processor-running-time-type": "system-determined"
}
```

Figure 279. Get Reset Activation Profile Properties: Response

Update Reset Activation Profile Properties

The **Update Reset Activation Profile Properties** operation updates one or more writeable properties of the Reset Activation Profile designated by *{reset-activation-profile-name}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/reset-activation-profiles/{*reset-activation-profile-name*}

URI variables:

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{reset-activation-profile-name}	Reset Activation Profile name

Request body contents

The request body is expected to contain one or more field names representing writable Reset Activation Profile properties, along with the new values for those fields.

The response body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the Reset Activation Profile to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the Reset Activation Profile is updated with the value provided by the input field, and status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **Customize/Delete Activation Profiles** task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	300	The provided update values would result in an illegal state. Verify that the values are both internally consistent and consistent with the current state of the profile.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{reset-activation-profile-name}*) does not designate an existing activation profile.
409 (Conflict)	2	The operation was rejected by the Support Element (SE), because the SE is currently performing processing that requires exclusive control of the SE. Retry the operation at a later time.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Image activation profile

An Image activation profile is used by an Activate operation to activate a logical partition of a previously activated CPC.

An activation profile can only be created or deleted from the Hardware Management Console or the Support Element.

See the *Support Element Operations Guide* for details on customizing activation profiles.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This element includes the following type-specific properties. Some properties have additional notes associated with them. Refer to the table notes at the end of this table.

Table 142. Image activation profile: type-specific properties

Name	Qualifier	Type	Description
element-uri	—	String/URI	The canonical URI path of the Image Activation Profile object, of the form `/api/cpcs/{cpc-id}/image-activation-profiles/{image-activation-profile-name}` where *{image-activation-profile-name}* is the value of the name property (Image Activation Profile name).
parent	—	String/URI	The canonical URI path of the associated CPC object.
class	—	String	The class of an Image Activation Profile object is **"image-activation-profile"**.
name	—	String (0-16)	The activation profile name, which uniquely identifies this profile within the set of activation profiles for the CPC object designated by *{cpc-id}*.
description	(w)	String (0-50)	The activation profile description
ipl-address[15]	(w)	String (0-5)	The hexadecimal address of an I/O device that provides access to the control program to be loaded. The input value will be right justified and padded with zeros to 5 characters. An empty string indicates that the value for this property is to be retrieved from the IOCDS used during a subsequent Load operation. Valid values are in the range "00000" to "nFFFF" where "n" is the number of subchannel sets provided by the CPC minus 1. So, for example, on a CPC that provides 3 subchannel sets, the valid range is "00000" to "2FFFF".
ipl-parameter	(w)	String (0-8)	Some control programs support the use of this property to provide additional control over the outcome of a Load operation. Refer to the configuration documentation for the control program to be loaded to see if this parameter is supported and if so, what values and format is supported. An empty string indicates that the value for this property is to be retrieved from the IOCDS used during a subsequent Load operation. Valid characters are 0-9, A-Z, blank and period. On an Update, a non-empty string is left justified and right padded with blanks to 8 characters.
initial-processing-weight[1]	(w)	Integer	The relative amount of shared general purpose processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the relative amount of shared general purpose processor resources initially allocated to the logical partition. Update: **1-999** Define the relative amount of shared general purpose processor resources allocated to the logical partition.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
initial-processing-weight-capped[1, 2, 3]	(w)	Boolean	Whether the initial processing weight for general purpose processors is a limit or a target. **True:** Indicates that the initial general purpose processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of general purpose processor resources. **False:** Indicates that the initial general purpose processor processing weight for the logical partition is not capped. It represents the share of general purpose processor resources guaranteed to a logical partition when all general purpose processor resources are in use. Otherwise, when excess general purpose processor resources are available, the logical partition can use them if necessary.
minimum-processing-weight[1]	(w)	Integer	The minimum relative amount of shared general purpose processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the minimum relative amount of shared general purpose processor resources allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared general purpose processor resources allocated to the logical partition. The value must be less than or equal to the initial-processing-weight property.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
maximum-processing-weight[1]	(w)	Integer	The maximum relative amount of shared general purpose processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared general purpose processor. **1-999** Represents the maximum relative amount of shared general purpose processor resources allocated to the logical partition. Update: **1-999** Defines the maximum relative amount of shared general purpose processor resources allocated to the logical partition. Must be greater than or equal to the initial-processing-weight property.
workload-manager-enabled[4]	(w)	Boolean	Whether or not z/OS Workload Manager is allowed to change processing weight related properties. **True:** Indicates that z/OS Workload Manager is allowed to change processing weight related properties for this logical partition. **False:** Indicates that z/OS Workload Manager is not allowed to change processing weight related properties for this logical partition.
defined-capacity	(w)	Integer	The defined capacity expressed in terms of Millions of Service Units (MSU)s per hour. MSU is a measure of processor resource consumption. The amount of MSUs a logical partition consumes is dependent on the model, the number of logical processors available to the partition, and the amount of time the logical partition is dispatched. The defined capacity value specifies how much capacity the logical partition is to be managed to by z/OS Workload Manager for the purpose of software pricing. **0:** No defined capacity is specified for this logical partition. **1-nnnn:** Represents the amount of defined capacity specified for this logical partition
ipl-type	(w)	String Enum	One of: • **"ipltype-standard"** - This image activation profile is used to perform a standard load. • **"ipltype-scsi"** - This image activation profile is used to perform a SCSI load. • **"ipltype-scsidump"** - This image activation profile is used to perform a SCSI dump.
worldwide-port-name[15]	(w)	String (1-16)	Worldwide port name of the target SCSI device, used for a SCSI load or SCSI dump, in hexadecimal.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
disk-partition-id	(w)	Integer (0-30)	The disk partition number (also called the boot program selector) for the activation profile, used for a SCSI load or SCSI dump.
logical-unit-number[15]	(w)	String (1-16)	Logical unit number value for the activation profile, used for a SCSI load or SCSI dump, in hexadecimal.
boot-record-lba[15]	(w)	String (1-16)	Boot record logical block address for the activation profile, used for a SCSI load or SCSI dump, in hexadecimal.
os-specific-load-parameters	(w)	String (0-256)	Operating system-specific load parameters for the activation profile, used for a SCSI load or SCSI dump. On an Update, value is left justified and right padded with blanks to 256 characters.
initial-aap-processing-weight[5]	(w)	Integer	The relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition at activation. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the relative amount of shared Application Assist Processor (zAAP) processor resources initially allocated to the logical partition. Update: **1-999** Define the relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.
initial-aap-processing-weight-capped[3, 5, 6]	(w)	Boolean	Whether the initial processing weight for Application Assist Processor (zAAP) processors is a limit or a target. **True:** Indicates that the initial Application Assist Processor (zAAP) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Application Assist Processor (zAAP) processor resources, regardless of the availability of excess Application Assist Processor (zAAP) processor resources. **False:** Indicates that the initial Application Assist Processor (zAAP) processor processing weight for the logical partition is not capped. It represents the share of Application Assist Processor (zAAP) processor resources guaranteed to a logical partition when all Application Assist Processor (zAAP) processor resources are in use. Otherwise, when excess Application Assist Processor (zAAP) processor resources are available, the logical partition can use them if necessary.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
minimum-aap-processing-weight[5]	(w)	Integer	The minimum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the minimum relative amount of shared Application Assist Processor (zAAP) processor resources initially allocated to the logical partition. Update: **0** No minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.
maximum-aap-processing-weight[5]	(w)	Integer	The maximum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor. **1-999** Represents the maximum relative amount of shared Application Assist Processor (zAAP) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Application Assist Processor (zAAP) processor resources allocated to the logical partition.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
initial-ifl-processing-weight[7]	(w)	Integer	The relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition at activation. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the logical partition. Update: **1-999** Define the relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.
initial-ifl-processing-weight-capped[3, 7, 8]	(w)	Boolean	Whether the initial processing weight for Integrated Facility for Linux (IFL) processors is a limit or a target. **True:** Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Integrated Facility for Linux (IFL) processor resources, regardless of the availability of excess Integrated Facility for Linux (IFL) processor resources. **False:** Indicates that the initial Integrated Facility for Linux (IFL) processor processing weight for the logical partition is not capped. It represents the share of Integrated Facility for Linux (IFL) processor resources guaranteed to a logical partition when all Integrated Facility for Linux (IFL) processor resources are in use. Otherwise, when excess Integrated Facility for Linux (IFL) processor resources are available, the logical partition can use them if necessary.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
minimum-ifl-processing-weight[7]	(w)	Integer	The minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.
maximum-ifl-pocessing-weight[7]	(w)	Integer	The maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition. Get: **0** The Image Activation profile does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor. **1-999** Represents the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Integrated Facility for Linux (IFL) processor resources allocated to the logical partition.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
initial-internal-cf-processing-weight[9]	(w)	Integer	The relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition at activation. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the logical partition. Update: **1-999** Define the relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition.
initial-internal-cf-processing-weight-capped[3, 9, 10]	(w)	Boolean	Indicates whether the initial processing weight for Internal Coupling Facility (ICF) processors is a limit or a target. **True:** Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the Image associated with the logical partition is capped. It represents the logical partition's maximum share of Internal Coupling Facility (ICF) processor resources, regardless of the availability of excess Internal Coupling Facility (ICF) processor resources. **False:** Indicates that the initial Internal Coupling Facility (ICF) processor processing weight for the logical partition is not capped. It represents the share of Internal Coupling Facility (ICF) processor resources guaranteed to a logical partition when all Internal Coupling Facility (ICF) processor resources are in use. Otherwise, when excess Internal Coupling Facility (ICF) processor resources are available, the logical partition can use them if necessary.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
minimum-internal-cf-processing-weight[9]	(w)	Integer	The minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition at activation. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor. **1-999** Represents the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the logical partition. Update: **1-999** Define the minimum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition.
maximum-internal-cf-processing-weight[9]	(w)	Integer	The maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor **1-999** Represents the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Internal Coupling Facility (ICF) processor resources allocated to the logical partition.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
initial-ziip-processing-weight[11]	(w)	Integer	The relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition at activation. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the logical partition. Update: **1-999** Define the relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
initial-ziip-processing-weight-capped[3, 11, 12]	(w)	Boolean	Whether the initial processing weight for Integrated Information Processors (zIIP) processors is a limit or a target. **True:** Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the logical partition is capped. It represents the logical partition's maximum share of Integrated Information Processors (zIIP) processor resources, regardless of the availability of excess Integrated Information Processors (zIIP) processor resources. **False:** Indicates that the initial Integrated Information Processors (zIIP) processor processing weight for the logical partition is not capped. It represents the share of Integrated Information Processors (zIIP) processor resources guaranteed to a logical partition when all Integrated Information Processors (zIIP) processor resources are in use. Otherwise, when excess Integrated Information Processors (zIIP) processor resources are available, the logical partition can use them if necessary.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
minimum-ziip-processing-weight[11]	(w)	Integer	The minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the logical partition. Update: **0** There is no minimum value for the processing weight. **1-999** Define the minimum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
maximum-ziip-processing-weight[11]	(w)	Integer	The maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition. Get: **0** The Image Activation Profile does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor. **1-999** Represents the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources initially allocated to the logical partition. Update: **1-999** Define the maximum relative amount of shared Integrated Information Processors (zIIP) processor resources allocated to the logical partition.
group-profile-uri	(w)	String/ URI	The canonical URI of the Group profile to be used for the logical partition activated by this profile, which provides the group capacity value. On a Get, a null object is returned if no Group profile is associated with this activation profile. On an Update, a null object indicates that no Group profile is to be associated with this activation profile.
load-at-activation	(w)	Boolean	If true, the logical partition will be loaded at the end of the activation.
central-storage	(w)	Integer	Defines the amount of central storage, measured in megabytes (MB), to be allocated for the logical partition's exclusive use at activation. This value must be a multiple of the storage granularity value.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
reserved-central-storage	(w)	Integer	Defines the amount of central storage, measured in megabytes (MB), dynamically reconfigurable to the logical partition after activation. This value must be a multiple of the storage granularity value.
expanded-storage	(w)	Integer	Defines the amount of expanded storage, measured in megabytes (MB), to be allocated for the logical partition's exclusive use at activation. This value must be a multiple of the storage granularity value.
reserved-expanded-storage	(w)	Integer	Defines the amount of expanded storage, measured in megabytes (MB), dynamically reconfigurable to the logical partition after activation. This value must be a multiple of the storage granularity value.
processor-usage	(w)	String Enum	Defines how processors are allocated to the logical partition. One of the following values: • **"dedicated"** - all processor types in the logical partition are to be exclusively available to this specific logical partition. • **"shared"** - all processors types in the logical partition are to be shareable across logical partitions.
number-dedicated-general-purpose-processors[13]	(w)	Integer	Defines the number of general purpose processors to be allocated for the logical partition's exclusive use at activation.
number-reserved-dedicated-general-purpose-processors[13]	(w)	Integer	Defines the number of dedicated general purpose processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-dedicated-aap-processors[13]	(w)	Integer	Defines the number of Application Assist Processor (zAAP) processors to be allocated for the logical partition partition's exclusive use at activation.
number-reserved-dedicated-aap-processors[13]	(w)	Integer	Defines the number of dedicated Application Assist Processor (zAAP) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-dedicated-ifl-processors[13]	(w)	Integer	Defines the number of Integrated Facility for Linux (IFL) processors to be allocated for the logical partition partition's exclusive use at activation.
number-reserved-dedicated-ifl-processors[13]	(w)	Integer	Defines the number of dedicated Integrated Facility for Linux (IFL) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-dedicated-icf-processors[13]	(w)	Integer	Defines the number of Integrated Coupling Facility (ICF) processors to be allocated for the logical partition partition's exclusive use at activation.
number-reserved-dedicated-icf-processors[13]	(w)	Integer	Defines the number of dedicated Integrated Coupling Facility (ICF) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-dedicated-ziip-processors[13]	(w)	Integer	Defines the number of Integrated Information Processors (zIIP) processors to be allocated for the logical partition partition's exclusive use at activation.
number-reserved-dedicated-ziip-processors[13]	(w)	Integer	Defines the number of dedicated Integrated Information Processors (zIIP) processors to be reserved for the logical partition, which can be dynamically configured after activation.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
number-shared-general-purpose-processors[14]	(w)	Integer	Defines the number of shared general purpose processors to be allocated for the logical partition at activation.
number-reserved-shared-general-purpose-processors[14]	(w)	Integer	Defines the number of shared general purpose processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-shared-aap-processors[14]	(w)	Integer	Defines the number of shared Application Assist Processor (zAAP) processors to be allocated for the logical partition at activation.
number-reserved-shared-aap-processors[14]	(w)	Integer	Defines the number of shared Application Assist Processor (zAAP) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-shared-ifl-processors[14]	(w)	Integer	Defines the number of shared Integrated Facility for Linux (IFL) processors to be allocated for the logical partition at activation.
number-reserved-shared-ifl-processors[14]	(w)	Integer	Defines the number of shared Integrated Facility for Linux (IFL) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-shared-icf-processors[14]	(w)	Integer	Defines the number of shared Integrated Coupling Facility (ICF) processors to be allocated for the logical partition at activation.
number-reserved-shared-ifl-processors[14]	(w)	Integer	Defines the number of shared Integrated Coupling Facility (ICF) processors to be reserved for the logical partition, which can be dynamically configured after activation.
number-shared-ziip-processors[14]	(w)	Integer	Defines the number of shared Integrated Information Processors (zIIP) processors to be allocated for the logical partition at activation.
number-reserved-shared-ziip-processors[14]	(w)	Integer	Defines the number of shared Integrated Information Processors (zIIP) processors to be reserved for the logical partition, which can be dynamically configured after activation.
basic-cpu-counter-authorization-control	(w)	Boolean	If true, the basic CPU counter facility for the logical partition is enabled.
problem-state-cpu-counter-authorization-control	(w)	Boolean	If true, the problem state CPU counter facility for the logical partition is enabled.
crypto-activity-cpu-counter-authorization-control	(w)	Boolean	If true, the crypto activity CPU counter facility for the logical partition is enabled.
extended-cpu-counter-authorization-control	(w)	Boolean	If true, the extended CPU counter facility for the logical partition is enabled.
coprocessor-cpu-counter-authorization-control	(w)	Boolean	If true, the coprocessor group CPU counter facility for the logical partition is enabled.
basic-cpu-sampling-authorization-control	(w)	Boolean	If true, the basic CPU sampling facility for the logical partition is enabled.
permit-aes-key-import-functions	(w)	Boolean	If true, importing of AES keys for the logical partition is enabled.
permit-des-key-import-functions	(w)	Boolean	If true, importing of DES keys for the logical partition is enabled.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
liccc-validation-enabled	(w)	Boolean	If true, ensure that the image profile data conforms to the current maximum Licensed Internal Code Configuration Control (LICCC) configuration.
global-performance-data-authorization-control	(w)	Boolean	If true, the logical partition can be used to view the processing unit activity data for all other logical partitions activated on the same CPC.
io-configuration-authorization-control	(w)	Boolean	If true, the logical partition can be used to read and write any Input/Output Configuration Data Set (IOCDS) in the configuration.
cross-partition-authority-authorization-control	—	Boolean	If true, the logical partition can be used to issue control program instructions that reset or deactivate other logical partitions.
logical-partition-isolation-control	(w)	Boolean	If true, reconfigurable channel paths assigned to the logical partition are reserved for its exclusive use.
operating-mode	(w)	String Enum	The operating mode for the logical partition: • **"esa390"** • **"esa390-tpf"** • **"coupling-facility"** • **"linux-only"** • **"zvm"** • **"zaware"**
clock-type	(w)	String Enum	One of: • **"standard"** – Set the logical partition's clock is set to the same time set for the CPC's time source. • **"lpar"** - Set the logical partition's clock using an offset from the External Time Source's time of day.
time-offset-days	(w)	Integer (0-999)	The number of days the logical partition's clock is to be offset from the External Time Source's time of day.
time-offset-hours	(w)	Integer (0-23)	The number of hours the logical partition's clock is to be offset from the External Time Source's time of day.
time-offset-minutes	(w)	Integer Enum	The number of minutes the logical partition⌂s clock is to be offset from the External Time Source's time of day. Allowable values are 0, 15, 30 or 45.
time-offset-increase-decrease	(w)	String Enum	One of: • **"increase"** – Set the logical partition's clock ahead of the External Time Source's time of day. • **"decrease"** – Set the logical partition's clock back from the External Time Source's time of day.
zaware-host-name[16]	(w)	String (1-64)	The IBM zAware host name. Valid characters are: a-z,A-Z,0-9, period(.), minus(-) and colon(:)
zaware-master-userid[16]	(w)	String (1-32)	The IBM zAware master userid. Valid characters are: a-z,A-Z,0-9, period(.), minus(-) and underscore (_)
zaware-master-pw[16]	(w)	String (8-256)	The IBM zAware master password. Valid characters are: a-z,A-Z,0-9 and !@#$%^&*()_+{}|<>?-= This property is not returned on a Get request, it can only be specified on an Update request.

Table 142. Image activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description
zaware-network-info[16]	(w)	Array of zaware-network objects	The set of networks available to IBM zAware. A maximum of 100 networks are permitted. On an Update request, this property fully replaces the existing set.
zaware-gateway-info[16]	(w)	ip-info object	The default gateway IP address information.
zaware-dns-info[16]	(w)	Array of ip-info objects	The DNS IP address information. A minimum of 0 entries and a maximum of 2 entries are permitted.

Notes:

1. An Update of this property is only valid for an Image Activation Profile that represents a logical partition with at least one shared general purpose processor.
2. The value returned for a Get request is always false when the Image Activation Profile does not represent a logical partition or the Image Activation Profile does not represent a logical partition with at least one shared general purpose processor.
3. This property and the **workload-manager-enabled** property are mutually exclusive and cannot both be enabled at the same time. Therefore in order to enable this property it might be necessary to first disable the **workload-manager-enabled** property.
4. This property and the various capping properties are mutually exclusive and cannot be enabled at the same time. Therefore in order to enable this property it may be necessary to first disable any capping property that is currently enabled.
5. An Update of this property is only valid for an Image Activation Profile that represents a logical partition with at least one shared Application Assist Processor (zAAP) processor.
6. The value returned for a Get request is always false when the Image Activation Profile does not represent a logical partition with at least one shared Application Assist Processor (zAAP) processor.
7. An Update of this property is only valid for an Image Activation Profile that represents a logical partition with at least one shared Integrated Facility for Linux (IFL) processor.
8. The value returned for a Get request is always false when the Image Activation Profile does not represent a logical partition with at least one shared Integrated Facility for Linux (IFL) processor.
9. An Update of this property is only valid for an Image Activation Profile that represents a logical partition with at least one shared Internal Coupling Facility (ICF) processor.
10. The value returned for a Get request is always false when the Image Activation Profile does not represent a logical partition with at least one shared Internal Coupling Facility (ICF) processor.
11. An Update of this property is only valid for an Image Activation Profile that represents a logical partition with at least one shared Integrated Information Processors (zIIP) processor.
12. The value returned for a GET request is always false when the Image Activation profile does not represent a logical partition with at least one shared Integrated Information Processors (zIIP) processor.
13. The value of this property is a null object if the processor-usage property is **"shared"**
14. The value of this property is a null object if the processor-usage property is **"dedicated"**
15. An Update request accepts any mixture of [a-f,A-F,0-9], however the original string value is not saved and a subsequent Get request may not return the exact same set of lower/upper case letters.
16. On a Get request, this property is returned only when **activation-mode** is **"zaware"**. On an Update request, this property can be specified only when **activation-mode** is **"zaware"**.

List Image Activation Profiles

The **List Image Activation Profiles** operation lists the Image Activation Profiles for the associated CPC object.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/image-activation-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`image-activation-profiles`	Array of image-actprof-info objects	Array of nested objects (described in the following table).

Each image-actprof-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the Image Activation Profile object.
`name`	String	The name of the Image Activation Profile.

Description

This operation lists the Image Activation Profiles associated with a particular CPC.

If the **name** query parameter is specified, the returned list is limited to those Image Activation Profiles that have a name property matching the specified filter pattern. If the **name** parameter is omitted, this filtering is not done.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	281	An unexpected error occurred during the collection of the list of activation profiles.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 280. List Image Activation Profiles: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:18 GMT
content-type: application/json;charset=UTF-8
content-length: 506
{
   "image-activation-profiles": [
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles/
        APIVM1",
         "name": "APIVM1"
      },
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles/
        DEFAULT",
         "name": "DEFAULT"
      },
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles/
        ZOS1",
         "name": "ZOS1"
      }
   ]
}
```

Figure 281. List Image Activation Profiles: Response

Get Image Activation Profile Properties

The **Get Image Activation Profile Properties** operation retrieves the properties of a single Image Activation Profile designated by *{image-activation-profile-name}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/image-activation-profiles/{*image-activation-profile-name*}

URI variables:

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{image-activation-profile-name}	Image Activation Profile name

Response body contents

On successful completion, the response body provides the current values of the properties for the Image Activation Profile as defined in the "Data model" on page 594.

Description

The URI path must designate an existing Image Activation Profile and the API user must have object-access permission to the associated CPC object. If either of these conditions is not met, HTTP status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined by the Data Model for the Image Activation Profile object.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{image-activation-profile-name}*) does not designate an existing activation profile.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles/ZOS HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 282. Get Image Activation Profile Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:18 GMT
content-type: application/json;charset=UTF-8
content-length: 3352
{
   "basic-cpu-counter-authorization-control": false,
   "basic-cpu-sampling-authorization-control": false,
   "boot-record-lba": "0",
   "central-storage": 4096,
   "class": "image-activation-profile",
   "clock-type": "standard",
   "coprocessor-cpu-counter-authorization-control": false,
   "cross-partition-authority-authorization-control": false,
   "crypto-activity-cpu-counter-authorization-control": false,
   "defined-capacity": 0,
   "description": "This is the ZOS Image profile.",
   "disk-partition-id": 0,
   "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/image-activation-profiles/ZOS",
   "expanded-storage": 0,
   "extended-cpu-counter-authorization-control": false,
   "global-performance-data-authorization-control": true,
   "group-profile-uri": null,
   "initial-aap-processing-weight": 0,
   "initial-aap-processing-weight-capped": false,
   "initial-ifl-processing-weight": 0,
   "initial-ifl-processing-weight-capped": false,
   "initial-internal-cf-processing-weight": 0,
   "initial-internal-cf-processing-weight-capped": false,
   "initial-processing-weight": 44,
   "initial-processing-weight-capped": false,
   "initial-ziip-processing-weight": 0,
   "initial-ziip-processing-weight-capped": false,
   "io-configuration-authorization-control": true,
   "ipl-address": "00000",
   "ipl-parameter": "        ",
   "ipl-type": "ipltype-standard",
   "liccc-validation-enabled": true,
   "load-at-activation": false,
   "logical-partition-isolation-control": false,
   "logical-unit-number": "0",
   "maximum-aap-processing-weight": 0,
   "maximum-ifl-processing-weight": 0,
   "maximum-internal-cf-processing-weight": 0,
   "maximum-processing-weight": 44,
```

Figure 283. Get Image Activation Profile Properties: Response (Part 1)

```
   "maximum-ziip-processing-weight": 0,
   "minimum-aap-processing-weight": 0,
   "minimum-ifl-processing-weight": 0,
   "minimum-internal-cf-processing-weight": 0,
   "minimum-processing-weight": 44,
   "minimum-ziip-processing-weight": 0,
   "name": "ZOS",
   "number-dedicated-aap-processors": null,
   "number-dedicated-general-purpose-processors": null,
   "number-dedicated-icf-processors": null,
   "number-dedicated-ifl-processors": null,
   "number-dedicated-ziip-processors": null,
   "number-reserved-dedicated-aap-processors": null,
   "number-reserved-dedicated-general-purpose-processors": null,
   "number-reserved-dedicated-icf-processors": null,
   "number-reserved-dedicated-ifl-processors": null,
   "number-reserved-dedicated-ziip-processors": null,
   "number-reserved-shared-aap-processors": 0,
   "number-reserved-shared-general-purpose-processors": 0,
   "number-reserved-shared-icf-processors": 0,
   "number-reserved-shared-ifl-processors": 0,
   "number-reserved-shared-ziip-processors": 0,
   "number-shared-aap-processors": 0,
   "number-shared-general-purpose-processors": 1,
   "number-shared-icf-processors": 0,
   "number-shared-ifl-processors": 0,
   "number-shared-ziip-processors": 0,
   "operating-mode": "esa390",
   "os-specific-load-parameters": "
          ",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "permit-aes-key-import-functions": true,
   "permit-des-key-import-functions": true,
   "problem-state-cpu-counter-authorization-control": false,
   "processor-usage": "shared",
   "reserved-central-storage": 0,
   "reserved-expanded-storage": 0,
   "time-offset-days": 0,
   "time-offset-hours": 0,
   "time-offset-increase-decrease": "decrease",
   "time-offset-minutes": 0,
   "workload-manager-enabled": false,
   "worldwide-port-name": "0"
}
```


Figure 284. Get Image Activation Profile Properties: Response (Part 2)

Update Image Activation Profile Properties

The **Update Image Activation Profile Properties** operation updates one or more writeable properties of the Image Activation Profile designated by *{image-activation-profile-name}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/image-activation-profiles/{*image-activation-profile-name*}

URI variables:

Variable	Description
{cpc-id}	Object ID of the target CPC object.

Variable	Description
{image-activation-profile-name}	Image Activation Profile name

Response body contents

The request body is expected to contain one or more field names representing writable Image Activation Profile properties, along with the new values for those fields.

The response body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the Image Activation Profile to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, HTTP status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the Image Activation Profile is updated with the value provided by the input field, and HTTP status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the Customize/Delete Activation Profiles task.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
	300	The provided update values would result in an illegal state. Verify that the values are both internally consistent and consistent with the current state of the profile.
	306	The provided update values are not valid for the current operating-mode, the target image activation profile's operating-mode is not **"zaware"** or was not specified as **"zaware"** in the request body.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	2	A URI in the request body does not designate an existing resource of the expected type, or designates a resource for which the API user does not have object-access permission.
	260	The activation profile name in the URI (*{image-activation-profile-name}*) does not designate an existing activation profile.
409 (Conflict)	2	The operation was rejected by the Support Element (SE), because the SE is currently performing processing that requires exclusive control of the SE. Retry the operation at a later time.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Load activation profile

A Load activation profile is used to load a previously activated logical partition with a control program or operating system.

An activation profile can only be created or deleted from the Hardware Management Console or the Support Element.

See the *Support Element Operations Guide* for details on customizing activation profiles.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This element includes the following type-specific properties.

Table 143. Load activation profile: type-specific properties

Name	Qualifier	Type	Description of specialization
element-uri	—	String/ URI	The canonical URI path of the Load Activation Profile object, of the form `/api/cpcs/{cpc-id}/load-activation-profiles/{load-activation-profile-name}` where *{load-activation-profile-name}* is the value of the name property (Load Activation Profile name).
parent	—	String/ URI	The canonical URI path of the associated CPC object
class	—	String	The class of a Load Activation Profile object is **"load-activation-profile"**.
name	—	String (1-16)	The activation profile name, which uniquely identifies this profile within the set of activation profiles for the CPC object designated by *{cpc-id}*.
description	(w)	String (1-50)	The load profile description

Table 143. Load activation profile: type-specific properties (continued)

Name	Qualifier	Type	Description of specialization
ipl-address	(w)	String (0-5)	The hexadecimal address of an I/O device that provides access to the control program to be loaded. The input value will be right justified and padded with zeros to 5 characters. An empty string indicates that the value for this property is to be retrieved from the IOCDS used during a subsequent Load operation. Valid values are in the range "00000" to "nFFFF" where "n" is the number of subchannel sets provided by the CPC minus 1. So, for example, on a CPC that provides 3 subchannel sets, the valid range is "00000" to "2FFFF".
ipl-parameter[1]	(w)	String (0-8)	Some control programs support the use of this property to provide additional control over the outcome of a Load operation. Refer to the configuration documentation for the control program to be loaded to see if this parameter is supported and if so, what values and format is supported. An empty string indicates that the value for this property is to be retrieved from the IOCDS used during a subsequent Load operation. On an Update, a non-empty string is left justified and right padded with blanks to 8 characters.
ipl-type	(w)	String Enum	One of: • **"ipltype-standard"** - Associated image activation profile is used to perform a standard load. • **"ipltype-scsi"** - Associated image activation profile is used to perform a SCSI load. • **"ipltype-scsidump"** - Associated image activation profile is used to perform a SCSI dump.
worldwide-port-name[1]	(w)	String (1-16)	Worldwide port name value for the activation profile, used for SCIS load and SCSI dump, in hexadecimal.
disk-partition-id	(w)	Integer (0-30)	Disk partition number (also called the boot program selector) for the activation profile, used for SCIS load and SCSI dump.
logical-unit-number[1]	(w)	String (1-16)	Logical unit number value for the activation profile, used for SCIS load and SCSI dump, in hexadecimal.
boot-record-lba[1]	(w)	String (1-16)	Boot record logical block address for the activation profile, used for SCIS load and SCSI dump, in hexadecimal.
os-specific-load-parameters	(w)	String (0-256)	Operating system-specific load parameters for the activation profile, used for SCIS load and SCSI dump.
clear-indicator	(w)	Boolean	Whether memory should be cleared before performing the Load (true) or not cleared (false). The default is to clear memory before performing the Load. This property cannot be set to false when the ipl-type is SCSI load or SCSI dump.
store-status-indicator	(w)	Boolean	Whether the store status function should be invoked before performing the Load (true) or not (false). The default is not to store status before performing the Load. The store status function stores the current values of the processing unit timer, the clock comparator, the program status word, and the contents of the processor registers in their assigned absolute storage locations. This property cannot be set to true when the ipl-type is SCSI load or SCSI dump, or when the ipl-type is **"ipltype-standard"** and clear-indicator is true.

1. An Update request accepts any mixture of [a-f, A-F, 0-9], however the original string value is not saved and a subsequent Get request may not return the exact same set of lower/upper case letters.

List Load Activation Profiles

The **List Load Activation Profiles** operation lists the Load Activation Profiles for the associated CPC object.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/load-activation-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`load-activation-profiles`	Array of load-actprof-info objects	Array of nested objects (described in the next table).

Each load-actprof-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the Load Activation Profile object.
`name`	String	The name of the Load Activation Profile.

Description

This operation lists the Load Activation Profiles for the associated CPC object.

If the name query parameter is specified, the returned list is limited to those Load Activation Profiles that have a name property matching the specified filter pattern. If the name parameter is omitted, this filtering is not done.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents" on page 618.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	281	An unexpected error occurred during the collection of the list of activation profiles.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/load-activation-profiles HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 285. List Load Activation Profiles: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:19 GMT
content-type: application/json;charset=UTF-8
content-length: 363
{
   "load-activation-profiles": [
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/load-activation-profiles/
        DEFAULTLOAD",
         "name": "DEFAULTLOAD"
      },
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/load-activation-profiles/
        MODIFYL",
         "name": "MODIFYL"
      }
   ]
}
```

Figure 286. List Load Activation Profiles: Response

Get Load Activation Profile Properties

The **Get Load Activation Profile Properties** operation retrieves the properties of a single Load Activation Profile designated by *{load-activation-profile-name}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/load-activation-profiles/{*load-activation-profile-name*}

URI variables

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{load-activation-profile-name}	Load Activation Profile name.

Response body contents

On successful completion, the response body provides the current values of the properties for the Load Activation Profile as defined in the "Data model" on page 616.

Description

The URI path must designate an existing Load Activation Profile and the API user must have object-access permission to the associated CPC object. If either of these conditions is not met, status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined by the Data Model for Load Activation Profiles.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{load-activation-profile-name}*) does not designate an existing activation profile.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/load-activation-profiles/DEFAULTLOAD HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 287. Get Load Activation Profile Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:19 GMT
content-type: application/json;charset=UTF-8
content-length: 793
{
   "boot-record-lba": "abcdef0123456789",
   "class": "load-activation-profile",
   "clear-indicator": true,
   "description": "This is the default Load profile.",
   "disk-partition-id": 0,
   "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/load-activation-profiles/
    DEFAULTLOAD",
   "ipl-address": "00D00",
   "ipl-parameter": "        ",
   "ipl-type": "ipltype-scsi",
   "logical-unit-number": "0",
   "name": "DEFAULTLOAD",
   "os-specific-load-parameters": "
    ",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340",
   "store-status-indicator": false,
   "worldwide-port-name": "0"
}
```

Figure 288. Get Load Activation Profile Properties: Response

Update Load Activation Profile Properties

The **Update Load Activation Profile Properties** operation updates one or more writeable properties of the Load Activation Profile designated by *{load-activation-profile-name}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/load-activation-profiles/{*load-activation-profile-name*}

URI variables

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{load-activation-profile-name}	Load Activation Profile name.

Request body contents

The request body is expected to contain one or more field names representing writable Load Activation Profile properties, along with the new values for those fields.

The response body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the Load Activation Profile to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, HTTP status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the Load Activation Profile is updated with the value provided by the input field, and HTTP status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **Customize/Delete Activation Profiles** task

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	300	The provided update values would result in an illegal state. Verify that the values are both internally consistent and consistent with the current state of the profile.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{load-activation-profile-name}*) does not designate an existing activation profile.
409 (Conflict)	2	The operation was rejected by the Support Element (SE), because the SE is currently performing processing that requires exclusive control of the SE. Retry the operation at a later time.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Group profile

A Group profile is used to define the group capacity value for all logical partitions belonging to that group.

A logical partition becomes a member of a group profile by placing the group profile's URI in the image activation profile used to activate the logical partition.

A group profile can only be created or deleted from the Hardware Management Console or the Support Element.

See the *Support Element Operations Guide* for details on customizing activation profiles.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This element includes the following type-specific properties.

Table 144. Group profile: type-specific properties

Name	Qualifier	Type	Description of specialization
element-uri	—	String/ URI	The canonical URI path of the group profile object, of the form `/api/cpcs/{cpc-id}/group-profiles/{group-profile-name}` where *{group-profile-name}* is the value of the name property (group profile name).
parent	—	String/ URI	The canonical URI path of the associated CPC object
class	—	String	The class of a Group Profile object is **"group-profile"**.
name	—	String (1-16)	The group profile name, which uniquely identifies this profile within the set of activation profiles for the CPC object designated by *{cpc-id}*
description	(w)	String (1-50)	The group profile description
capacity	(w)	Integer	The upper bound, in MSUs, beyond which the rolling 4-hour average CPU utilization cannot exceed for the group. A value of 0 indicates the setting is unused.

List Group Profiles

The **List Group Profiles** operation lists the Group Profiles for the associated CPC object.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/group-profiles

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Query parameters:

Name	Type	Rqd/Opt	Description
name	String	Optional	A regular expression used to limit returned objects to those that have a matching name property. If matches are found, the response will be an array with all objects that match. If no match is found, the response will be an empty array.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`group-profiles`	Array of group-actprof-info objects	Array of nested objects (described in the next table).

Each group-actprof-info object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the Group Profile object.
`name`	String	The name of the Group Profile.

Description

This operation lists the Group Profiles for the associated CPC object.

If the name query parameter is specified, the returned list is limited to those Group Profiles that have a name property matching the specified filter pattern. If the name parameter is omitted, this filtering is not done.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	299	A query parameter has an invalid syntax.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	281	An unexpected error occurred during the collection of the list of group profiles.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/group-profiles HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 289. List Group Profiles: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:19 GMT
content-type: application/json;charset=UTF-8
content-length: 182
{
   "group-profiles": [
      {
         "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/group-profiles/DEFAULT",
         "name": "DEFAULT"
      }
   ]
}
```

Figure 290. List Group Profiles: Response

Get Group Profile Properties

The **Get Group Profile Properties** operation retrieves the properties of a single Group Profile designated by *{group-profile-name}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/group-profiles/{*group-profile-name*}

URI variables

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{group-profile-name}	Group Profile name.

Response body contents

On successful completion, the response body provides the current values of the properties for the Group Profile as defined in the "Data model" on page 623.

Description

The URI path must designate an existing Group Profile and the API user must have object-access permission to the associated CPC object. If either of these conditions is not met, HTTP status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined by the Data Model for the Group Profile.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{group-profile-name}*) does not designate an existing group profile.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
GET /api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/group-profiles/DEFAULT HTTP/1.1
x-api-session: 5obf0hwsfv1sg9kr5f93cph3zt6o5cptb6lcl538wuyebdyzu4
```

Figure 291. Get Group Profile Properties: Request

```
200 OK
server: zSeries management console API web server / 1.0
cache-control: no-cache
date: Fri, 25 Nov 2011 17:16:20 GMT
content-type: application/json;charset=UTF-8
content-length: 250
{
   "capacity": 0,
   "class": "group-profile",
   "description": "This is the default Group profile.",
   "element-uri": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340/group-profiles/DEFAULT",
   "name": "DEFAULT",
   "parent": "/api/cpcs/37c6f8a9-8d5e-3e5d-8466-be79e49dd340"
}
```

Figure 292. Get Group Profile Properties: Response

Update Group Profile Properties

The **Update Group Profile Properties** operation updates one or more writeable properties of the Group Profile object designated by *{group-profile-name}*.

HTTP method and URI

POST /api/cpcs/{*cpc-id*}/group-profiles/{*group-profile-name*}

URI variables

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{group-profile-name}	Group Profile name.

Request body contents

The request body is expected to contain one or more field names representing writable Group Profile properties, along with the new values for those fields.

The response body can and should omit fields for properties whose values are not to be changed by this operation. Properties for which no input value is provided remain unchanged by this operation.

Description

The request body object is validated against the data model for the Group Profile to ensure that the request body contains only writeable properties and the data types of those properties are as required. If the request body is not valid, HTTP status code 400 (Bad Request) is returned with a reason code indicating the validation error encountered.

On successful execution, the value of each corresponding property of the Group Profile is updated with the value provided by the input field, and HTTP status code 204 (No Content) is returned.

When this operation changes the value of any property for which property-change notifications are due, those notifications are emitted asynchronously to this operation.

Authorization requirements

This operation has the following authorization requirements:
- Object access permission to the CPC object designated by *{cpc-id}*
- Action/task permission for the **Customize/Delete Activation Profiles** task

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	300	The provided update values would result in an illegal state. Verify that the values are both internally consistent and consistent with the current state of the profile.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	260	The activation profile name in the URI (*{group-profile-name}*) does not designate an existing group profile.
409 (Conflict)	2	The operation was rejected by the Support Element (SE), because the SE is currently performing processing that requires exclusive control of the SE. Retry the operation at a later time.
500 (Server Error)	281	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Capacity records

A capacity record represents a temporary upgrade that can be applied to a CPC.

These upgrades are provided through the following offerings:
- **On/Off Capacity on Demand (On/Off CoD)** - This offering allows you to temporarily add additional capacity or specialty engines due to seasonal activities, period-end requirements, peaks in workload, or application testing.
- **Capacity Backup (CBU)** - This offering allows you to replace model capacity or specialty engines to a backup server in the event of an unforeseen loss of server capacity because of an emergency.
- **Capacity for Planned Events (CPE)** - This offering allows you to replace model capacity or specialty engines due to a relocation of workload during system migrations or a data center move.

Data model

For definitions of the qualifier abbreviations in the following tables, see "Property characteristics" on page 32.

This element includes the following type-specific properties.

Table 145. Capacity records: type-specific properties

Name	Type	Description
element-uri	String/ URI	The canonical URI path of the capacity record object, of the form `/api/cpcs/{cpc-id}/capacity-records/{capacity-record-id}` where *{cpc-id}* is the value of the **object-id** property of the CPC object and *{capacity-record-id}* is the value of the **record-identifier** property of the Capacity Record object.
parent	String/ URI	The canonical URI path for the associated CPC object
class	String	The class of a capacity record object is **"capacity-record"**.
record-identifier	String (1-8)	The identifier for the capacity record.
record-type	String Enum	The type of capacity record. One of: • **"unknown"** - the record does not specify a record-type • **"cbu"** - a Capacity Backup Upgrade record • **"oocod"** - an On/Off Capacity on Demand record • **"planned-event"** - a Capacity for Planned Events record • **"loaner"** - resources loaned to the installation.
activation-status	String Enum	An indication if any of the resources defined for the record are currently activated. One of: • **"unknown"** - the activation status of the record is not known • **"not-activated"** - the record is not currently active • **"real"** - the record is either active or pending activation, via an Add Temporary Capacity operation with a test=false option • **"test"** - the record is either active or pending activation via an Add Temporary Capacity operation with a test=true option • **"can-be-activated"** - the record is available for activation, but not currently active.
activation-date	Timestamp	Defines the time stamp for when additional capacity for the record was activated.
record-expiration-date	Timestamp	Defines the time stamp for when the capacity record will expire.
activation-expiration-date	Timestamp	Defines the time stamp for when the additional capacity activated for the record will expire and no longer be active.
maximum-real-days	Integer	Defines the maximum days that real additional capacity can be activated for the record. A value of -1 indicates that the number of days is unlimited.
maximum-test-days	Integer	Defines the maximum days that test additional capacity can be activated for the record. A value of -1 indicates that the number of days is unlimited.
remaining-real-days	Integer	Defines the remaining number of days that additional real capacity can be active for the record. A value of -1 indicates that the number of days is unlimited.
remaining-test-days	Integer	Defines the remaining number of days that additional test capacity can be active for the record. A value of -1 indicates that the number of days is unlimited.
remaining-number-of-real-activations	Integer	Defines the number of times that real additional capacity can be activated for the record. A value of -1 indicates that activation count is unlimited.
remaining-number-of-test-activations	Integer	Defines the number of times that test additional capacity can be activated for the record. A value of -1 indicates that activation count is unlimited.

Table 145. Capacity records: type-specific properties (continued)

Name	Type	Description
processor-info	Array of caprec-proc-info objects	A nested object describing the processor capacities available with this capacity record.
available-targets	Array of caprec-target objects	A nested object describing the set of possible activation and deactivation targets contained within this capacity record. One of these targets is chosen via the software-model request body field on the **Add Temporary Capacity** or **Remove Temporary Capacity** operations.

Table 146. caprec-proc-info object

Name	Type	Description
type	String Enum	Identifies the type of specialty processor represented. One of: • **"cp"** - central processor • **"aap"** - Application Assist Processor • **"ifl"** - Integrated Facility for Linux processor • **"icf"** - Internal Coupling Facility processor • **"iip"** - Integrated Information Processors processor • **"sap"** - System Assist Processor.
processor-step	Integer	The number of processors steps available
speed-step	Integer	The CP processor speed activation step. A null object is returned for all other processor types.
max-number-processors	Integer	The maximum number of processors available
remaining-processor-days	Integer	The remaining processor days for this processor type. A -1 indicates an unlimited number of days.
remaining-msu-days	Integer	The remaining MSU days for this processor type. A -1 indicates an unlimited number of days. A null object is returned for processor types where this field is not meaningful.

Table 147. caprec-target object

Name	Type	Description
processor-step	Integer	The CPU processor step. This is the incremental delta CPUs compared to the current activation level. The returned value may be negative.
speed-step	Integer	The CPU processor speed activation step. This is the incremental delta speed steps compared to current activation level. The returned value may be negative.
software-model	String (1-3)	The software model that this target represents
billable-msu-cost	Integer	The overall billable MSU cost for this target
billable-msu-delta	Integer	The change in billable MSU cost by activating this target. The value may be negative.

List Capacity Records

The **List Capacity Records** operation lists the capacity record for a given CPC that are managed by this HMC.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/capacity-records

In this request, the URI variable *{cpc-id}* is the object ID of the target CPC object.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
`capacity-record`	Array of objects	Array of nested objects (described in the next table).

Each nested object contains the following fields:

Field name	Type	Description
`element-uri`	String/URI	Canonical URI path of the Capacity Record object.
`record-identifier`	String	The record identifier of the Capacity Record

Description

This operation lists the capacity record for a given CPC that are managed by this HMC.

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section.

Authorization requirements

This operation has the following authorization requirement:
- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in "Response body contents."

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
500 (Server Error)	275	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Get Capacity Record Properties

The **Get Capacity Record Properties** operation retrieves the properties of a single Capacity Record designated by *{capacity-record-id}* from the CPC object designated by *{cpc-id}*.

HTTP method and URI

GET /api/cpcs/{*cpc-id*}/capacity-records/{*capacity-record-id*}

URI variables

Variable	Description
{cpc-id}	Object ID of the target CPC object.
{capacity-record-id}	Capacity Record identifier, returned by a previous List Capacity Records operation

Response body contents

On successful completion, HTTP status code 200 (OK) is returned and the response body provides the current values of the properties for the Capacity Record as defined in "Data model" on page 628.

Description

The URI path must designate an existing Capacity Record and the API user must have object-access permission to the associated CPC object. If either of these conditions is not met, status code 404 (Not Found) is returned.

On successful execution, HTTP status code 200 (OK) is returned and the response body contains all of the current properties as defined by the Data Model for the Capacity Record object.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to the CPC object designated by *{cpc-id}*.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in the Response Body Contents section.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
	276	The capacity record has expired, it can be deleted from the SE.
	302	The capacity record identifier in the URI (*{capacity-record-id}*) must be 1 to 8 characters.
404 (Not Found)	1	The object ID in the URI (*{cpc-id}*) does not designate an existing CPC object, or the API user does not have object access permission to the object.
	274	The capacity record identifier in the URI (*{capacity-record-id}*) does not designate an existing capacity record.
500 (Server Error)	275	An unexpected error occurred during the operation.
503 (Service Unavailable)	1	The request could not be processed because the HMC is not communicating with the SE needed to perform the requested operation.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Chapter 15. Inventory and metrics services

The functions described in this chapter are termed "services" because unlike the interfaces described in many of the other chapters of this document, the functions described here are service-oriented rather than object-oriented in nature. That is, the functions of these services operate across multiple instances of managed objects rather than being directed at particular managed object instances.

The Inventory Service provides an efficient mechanism for retrieving identify and configuration information about all of the manageable resource instances that are managed by zManager. It provides this information in bulk form via a single request, and thus is expected to be a much more efficient means of determining this information than walking the entire resource tree one object at a time. It is anticipated that this service supports the requirements of a "discovery" phase of a client application that is interested in configuration information about all resources managed by zManager.

The Metrics Service provides a mechanism to retrieve performance metric data for resources that are managed by zManager. This data is captured periodically and buffered on the HMC. The data may include snapshots of performance data at a moment in time, or accumulated performance data, or both, as appropriate. This service is designed to support client applications that provide monitoring function for zManager managed resources.

Inventory services operations summary

The following operation is provided by the Inventory service:

Table 148. Inventory service: operations summary

Operation name	HTTP method and URI path
"Get Inventory" on page 636	`POST /api/services/inventory`

Metrics service operations summary

The following operations are provided by the Metrics service:

Table 149. Metrics service: operations summary

Operation name	HTTP method and URI path
"Create Metrics Context" on page 642	`POST /api/services/metrics/context`
"Get Metrics" on page 645	`GET /api/services/metrics/context/{metrics-context-id}`
"Delete Metrics Context" on page 649	`DELETE /api/services/metrics/context/{metrics-context-id}`

Table 150. Metrics service: URI variables

Variable	Description
{metrics-context-id}	Identifier of a metrics context object. Metrics contexts are associated with API sessions. Thus, this identifier is assigned by the metrics service so that it is unique within an API session and has a lifetime scoped to that session.

Inventory service

The Inventory Service is an API which allows the client application to fetch a list of ensemble resources and their properties.

This service is intended to support clients that need to determine the inventory and properties of all of the resources of the ensemble (or at least a large portion of those resources). Retrieving this information in bulk form using this service is expected to be much more efficient than walking the resource tree one object at a time using the object-oriented operations of the Web Services API.

The ability to filter the results to only certain classes of resources is provided.

A response to an inventory request is a series of JSON objects returned using HTTP chunked transfer encoding. These objects will be in a format specified in the corresponding resource class's Inventory Data Model sections.

Resources returned are those to which the API client has object-level authorization.

Get Inventory

The **Get Inventory** operation fetches ensemble resources and associated properties.

HTTP method and URI

```
POST /api/services/inventory
```

Request body contents

The request body can include a specification of the classes of resources that should be returned. It contains the following field:

Field name	Type	Description
resources	Array of String Enum	An array of String values. Each element specifies a category or class of resource that should be returned. A category is simply a grouping of classes, so specifying a category is functionally equivalent to specifying all of its member classes. The request may include both categories and classes. Omitting the resources field, or providing an empty array, is equivalent to specifying an array listing all of the supported classes. Categories and associated class values: • Category: **"zvm-resources"** – Class: **"zvm-virtualization-host"** – Class: **"zvm-virtual-server"** • Category: **"power-vm-resources"** – Class: **"power-vm-virtualization-host"** – Class: **"power-vm-virtual-server"** • Category: **"x-hyp-resources"** – Class: **"x-hyp-virtualization-host"** – Class: **"x-hyp-virtual-server"** • Category: **"prsm-resources"** – Class: **"prsm-virtualization-host"** – Class: **"prsm-virtual-server"** • Category: **"virtual-server-common"** – Class: **"power-vm-virtual-server-common"** – Class: **"prsm-virtual-server-common"** – Class: **"x-hyp-virtual-server-common"** – Class: **"zvm-virtual-server-common"** • Category: **"zbx-resources"** – Class: **"zbx"** – Class: **"rack"** – Class: **"power-blade"** – Class: **"system-x-blade"** – Class: **"isaopt-blade"** – Class: **"dpxi50z-blade"** – Class: **"bladecenter"** • Category: **"ensemble-wide-resources"** – Class: **"ensemble"** – Class: **"workload-resource-group"** – Class: **"virtual-network"** – Class: **"storage-resource"** • Category: **"core-resources"** – Class: **"cpc"** – Class: **"logical-partition"** • Category: **"console-resources"** – Class: **"console"** – Class: **"custom-group"**

Notes:

- The various classes of virtual server, virtualization host, and blade above (for example, **"x-hyp-virtual-server"**, **"prsm-virtualization-host"**, and **"power-blade"**) are actually type-specific variations of the object classes **"virtual-server"**, **"virtualization-host"**, and **"blade"**. They are specified with type qualifiers in the names here to allow distinguishing these types on inventory queries. The objects returned in the inventory response will be of the actual object classes (**"virtual-server"**, **"virtualization-host"**, or **"blade"**), and will have appropriate type values as defined in the data models for those classes.
- The various classes within the "virtual-server-common" category above represent inventory queries that return a common subset of the virtual server's properties rather than the entire set of properties defined in the Data Model. They correspond to **Get Virtual Server Properties** requests with the **properties=common** query parameter specified. Refer to the documentation for the **Get Virtual Servers Properties** operation and the discussion of inventory service data for Virtual Server objects for specific information on the properties provided.

Response body contents

On successful completion, the response body is a JSON array of JSON objects sent using HTTP chunked transfer encoding. The order in which these objects are returned is unspecified.

The array element documents are of 2 types:
- For resources that were successfully inventoried, the document will be as specified in the corresponding resource's inventory service data.
- For resources that were found but not successfully fully inventoried (i.e. the Object URI can be determined but not the properties), an inventory error document will be returned. Note that, even if one or more of these inventory error documents is contained in the response, an HTTP status code of 200 (OK) is still returned. The fields in the inventory error document are:

Field name	Type	Description
`uri`	String/URI	Canonical URI of the resource which could not be fully inventoried.
`class`	String	The class for these error documents is **"inventory-error"**.
`inventory-error-code`	Integer	A reason code for the inventory failure. Note that all of these reasons indicate success in locating a resource, but some sort of failure in gathering its properties during inventory collection. A subsequent call to get the properties for the URI in this error document may succeed. • 1: Resource not found on target. Although the resource's URI was located on the HMC, its properties were subsequently not located on the HMC or SE on which the property data for the managed object is to be gathered. • 2: Communication problem. Communication problems were experienced with the SE on which the property data for the managed object is to be gathered. • 3: Plugin load error. The code responsible for capturing the properties of a resource class experienced an unexpected problem loading. • 4: Unknown plugin error. The code responsible for capturing the properties of a resource returned an unrecognized error. • 5: Unexpected plugin error. The code responsible for capturing the properties of a resource returned an unexpected error. • 6: Timeout error. The code responsible for capturing the properties of a resource did not respond within the time allocated to it.
`inventory-error-text`	String	An error description for the inventory failure.
`inventory-error-details`	inventory-error-info Object	A nested inventory-error-info object that provides additional diagnostic information for unexpected inventory plugin errors. This field is provided if the **inventory-error-code** field is 5 (indicating unexpected plugin error). It is not provided for other **inventory-error-code** values. The format of the inventory-error-info object is defined in the next table.

The inventory-error-info object contains the following fields:

Field name	Type	Description
http-status	Integer	HTTP status code for the request.
request-uri	String	The URI that caused this error response.
reason	Integer	Numeric reason code providing more details as to the nature of the error) than is provided by the HTTP status code itself. This reason code is treated as a sub-code of the HTTP status code and thus must be used in conjunction with the HTTP status code to determine the error condition. Standard reason codes that apply across the entire API are described in“Common request validation reason codes” on page 18. Additional operation-specific reason codes may also be documented in the description of the specific API operations.

Field name	Type	Description
message	String	Message describing the error. This message is not currently localized.
stack	String	Internal HMC diagnostic information for the error. This field is supplied only on selected 5xx HTTP status codes.
error-details	Object	A nested object that provides additional operation-specific error information. This field is provided by selected operations, and the format of the nested object is as described by that operation.

Description

The Get Inventory operation returns information on ensemble resources and associated properties.

A resource is included in the response if it matches any one of the list of resource classes in the request body. Specifying a category is equivalent to specifying its member classes. If a class is repeated on the request, either explicitly or effectively via categories, the operation will behave as if the class were only specified once. If no resources are specified in the request body, all resources are returned.

Furthermore, a resource is included in the response only if the API user has object-access permission for that resource. If an HMC is a manager of a resource but the API user does not have permission to it, that resource is simply omitted from the response. A success status code is still returned.

If the HMC does not manage any resources to which the user has access, or if no resources are found that match the request body specification, an empty response is returned with a 204 (No Content) status code.

In addition to objects for inventoried resources, the response may include objects for resources whose URIs could be determined, but whose properties could not, for some reason, be obtained. Rather than treat these resources as completely non-inventoried and omit them, the URI and an error reason are returned.

The order in which the objects are returned is unspecified.

The Get Inventory implementation may choose to limit the number of simultaneous in-process inventory requests. If such a limit is reached, further requests will return an HTTP 503 (Service Unavailable) error status code until previous requests complete and the number of in-process inventory requests falls back below the limit.

Authorization requirements

This operation has the following authorization requirement:

- Object access permission to any object to be included in the result.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in “Response body contents” on page 638. If there are no resources to provide, HTTP status code 204 (No Content) is returned, along with an empty response body.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.
503 (Service Unavailable)	200	The request could not be processed because of the number of currently pending inventory requests. The request can be reissued at a later time.

Additional standard status and reason codes can be returned, as described in Chapter 3, “Invoking API operations,” on page 13.

Usage notes

The **Get Inventory** results represent a snapshot of inventory results as viewed from the HMC. The actual inventory can change, even as the results are being streamed back to the API client. Therefore, if the client wishes to stay informed about changes to the inventory and not risk missing any inventory changes, it should use the API event mechanisms to subscribe to inventory-related events before even issuing a **Get Inventory** request.

The **Get Inventory** results do not reflect all properties at a single moment in time. During the overall inventory collection process multiple resource's states and other properties may change. Therefore, states (or other properties) among two or more resources that might normally be expected to match (or have some other expected relationship) at one moment in time may instead return apparently inconsistent results in the **Get Inventory** response.

Example HTTP interaction

The following example illustrates a typical response for a **Get Inventory** request for the ensemble class of resources. Responses for other classes will differ significantly from this because the data differs on a class by class basis. The format of the data returned by the Inventory Service for each class of object is described in a section entitled “Inventory Service Data” within the documentation for that object class.

```
POST /api/services/inventory HTTP/1.1
x-api-session: 38gu0i9so1y0vjemqyptpcadnwq0esu32pjd3ub4jy6lxadp72
content-type: application/json
content-length: 27
{
   "resources": [
      "ensemble"
   ]
}
```

Figure 293. Get Inventory: Request

```
200 OK
transfer-encoding: chunked
server: zSeries management console API web server / 1.0
cache-control: no-cache, no-cache
date: Wed, 07 Dec 2011 03:42:15 GMT
content-type: application/json;charset=UTF-8
[
   {
      "acceptable-status": [
         "alternate-communicating"
      ],
      "class": "ensemble",
      "cpu-perf-mgmt-enabled-power-vm": false,
      "cpu-perf-mgmt-enabled-zvm": false,
      "description": "long ensemble name",
      "has-unacceptable-status": false,
      "is-locked": false,
      "load-balancing-enabled": false,
      "load-balancing-ip-addresses": [],
      "load-balancing-port": 3860,
      "mac-prefix": "02:00:00:00:00:00",
      "management-enablement-level": "automate",
      "name": "HMC_R74_ENSEMBLE",
      "object-id": "1f7ffb02-de39-11e0-88bd-00215e67351a",
      "object-uri": "/api/ensembles/1f7ffb02-de39-11e0-88bd-00215e67351a",
      "parent": null,
      "power-consumption": 24474,
      "power-rating": 65644,
      "reserved-mac-address-prefixes": [],
      "status": "alternate-communicating",
      "unique-local-unified-prefix": "fd2c:34be:df2:0:0:0:0:0"
   },
   {
      "class": "node",
      "element-uri": "/api/ensembles/1f7ffb02-de39-11e0-88bd-00215e67351a/nodes/
      9ba3b1aa-693a-3408-80ae-9d0808147ffa",
      "member": "/api/cpcs/9ba3b1aa-693a-3408-80ae-9d0808147ffa",
      "parent": "/api/ensembles/1f7ffb02-de39-11e0-88bd-00215e67351a",
      "type": "cpc"
   }
]
```

Figure 294. Get Inventory: Response

Metrics service

The zEnterprise (or later) Ensembles, Central Processing Complexes (CPCs), and their associated system resources are instrumented at key points to collect performance and utilization data. The data is forwarded by the metric data providers to a buffer on the HMC where it is made available to consumers of this API.

The data collection instrumentation organizes and formalizes the data it collects into a series of named metric groups. The Metrics Service API allows specification of the metric groups the client wishes to query. The API returns some information about the format of the metrics that are being fetched. Specifically, a structure called a metrics context is associated with any metrics retrieval, and that structure includes metric group names, individual metric field names, and the associated individual metric data types.

The full metric group documentation, however, including descriptions of the data collected and the frequency of collection, can be found in Chapter 16, “zManager metric groups,” on page 651.

Create Metrics Context

The **Create Metrics Context** operation creates a context under which metrics can be repeatedly retrieved. This context will be associated with the API session under which it was created.

HTTP method and URI

POST /api/services/metrics/context

Request body contents

A request body must be specified. It has the following fields:

Field name	Type	Description
anticipated-frequency-seconds	Integer	The number of seconds the client anticipates will elapse between **Get Metrics** calls against this context. The minimum accepted value is 15.
metric-groups	Array of Strings	Optional. Array of metric group names. If specified, then results from future **Get Metrics** requests associated with this context will be limited to only metrics with group names matching one of the specified values. If not specified, or if an empty array is specified, then results will not be limited with respect to metric group names.

Response body contents

On successful completion, the response body contains a JSON object with the following fields:

Field name	Type	Description
metrics-context-uri	String/URI	Canonical URI path of the metrics context object created by this operation This includes the metrics-context-id. E.g. “/api/services/metrics/context/1”, where “1” is the metrics-context-id.
metric-group-infos	Array of objects	Array of metric-group-info objects (described in the next table) that describe the data format for each metric group that may be returned by future GETs associated with this metric context.

Each nested metric-group-info object contains the following fields:

Field name	Type	Description
group-name	String	The name of the metric group for which we are providing descriptive information.
metric-infos	Array	Array of metric-info objects (described in the next table). These describe each metric for the group in the order that they will appear in future GETs associated with this context.

Each nested metric-info object contains the following fields:

Field name	Type	Description
metric-name	String	The name of the metric.

Field name	Type	Description
metric-type	String Enum	One of the following, describing the type of the metric: **"boolean-metric"**, **"byte-metric"**, **"double-metric"**, **"long-metric"**, **"integer-metric"**, **"short-metric"**, **"string-metric"** See the Get Metrics "Response body contents" on page 646 for further information on these metric types.

Description

This operation establishes a context for future **Get Metrics** operations that is valid for the current API session. Because of the high frequency of invocation and large volume of data expected, the metrics service interface has been structured to optimize the transmission of data on each **Get Metrics** request. Thus, rather than use a self-describing representation for the results returned by each Get Metrics, the metrics service instead provides the descriptive metadata as results from this **Create Metrics Context** operation. It then returns the metric data in a compact format each time **Get Metrics** is invoked.

At a high level, the **Create Metrics Context** response communicates two primary pieces of information back to the client. One is the metrics-context-uri, which includes the ID of the metrics context that must be referenced on future GETs to associate them with this context. The other is the metric-groups description data. That data provides the metric type and metric name information for each metric group whose metrics may be returned by this context. This may be useful to the client for determining how to parse future **Get Metrics** responses for this context, although the full documentation on metric group formats is found in Chapter 16, "zManager metric groups," on page 651.

The anticipated-frequency-seconds specification which is required on the request body tells the metrics service how frequently the client anticipates issuing **Get Metrics** requests against this context. The metrics service may take no action based on this frequency, but reserves the right to invalidate and delete the metrics context if 4 times the specified frequency passes without receipt of an associated **Get Metrics** operation.

Optional result filtering is provided by field metric-groups on the request body. If a non-empty metric-groups arrays is specified, then future **Get Metrics** operations associated with this context will return only groups with names listed there.

Additionally, if a metric-groups array of group names is specified on the **Create Metrics Context** request, then the response JSON document will include only matching metric-group-info fields. If one or more names in the metric-groups array does not represent a metric group registered on the HMC, then HTTP error status code 400 (Bad Request) will be returned and the context will not be established.

Although the POST response fully describes and guarantees the ordering of metric-infos within a metric group for that context, as a matter of policy the HMC will further guarantee that, for a given metric group, any additions of new metrics to the group will be to the end of the list for the group.

Authorization requirements

There are no authorization restrictions on creating a metrics context. However any future metric results returned by **Get Metrics** queries against that context will be restricted to managed objects accessible according to the permissions associated with the API session under which the metrics context was established.

Note that there is no indication via an HTTP status or reason code that future results may be restricted due to authorization restrictions. Rather, success is indicated and future **Get Metrics** responses behave just as if any restricted objects did not exist.

HTTP status and reason codes

On success, HTTP status code 201 (Created) is returned and the response body is provided as described in "Response body contents" on page 642. The URI for the newly created context is also provided in the **Location** header of the response.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

Note: These are example rather than actual metrics group names.

```
POST /api/services/metrics/context HTTP/1.1
x-api-session: 6a9oz3ymut6rvjijrft0loqhfzgpp0rnu4mjishwh6d39jh31q
content-type: application/json
content-length: 96
{
   "anticipated-frequency-seconds": 45,
   "metric-groups": [
      "virtualization-host-cpu-memory-usage"
   ]
}
```

Figure 295. Create Metrics Context: Request

```
201 Created
transfer-encoding: chunked
server: zSeries management console API web server / 1.0
cache-control: no-cache, no-cache
date: Wed, 07 Dec 2011 04:01:59 GMT
content-type: application/json;charset=UTF-8
{
   "metric-group-infos": [
      {
         "group-name": "virtualization-host-cpu-memory-usage",
         "metric-infos": [
            {
               "metric-name": "processor-usage",
               "metric-type": "integer-metric"
            },
            {
               "metric-name": "memory-usage",
               "metric-type": "integer-metric"
            },
            {
               "metric-name": "network-usage",
               "metric-type": "integer-metric"
            },
            {
               "metric-name": "storage-rate",
               "metric-type": "integer-metric"
            },
            {
               "metric-name": "physical-cpu-time",
               "metric-type": "long-metric"
            },
            {
               "metric-name": "memory-used",
               "metric-type": "integer-metric"
            },
            {
               "metric-name": "virt-host-management-cpu-time-used",
               "metric-type": "long-metric"
            },
            {
               "metric-name": "virt-host-page-ins",
               "metric-type": "long-metric"
            },
            {
               "metric-name": "virt-host-page-outs",
               "metric-type": "long-metric"
            }
         ]
      }
   ],
   "metrics-context-uri": "/api/services/metrics/context/1"
}
```

Figure 296. Create Metrics Context: Response

Get Metrics

The **Get Metrics** operation retrieves the current set of metrics associated with an established metrics context.

HTTP method and URI

GET /api/services/metrics/context/{*metrics-context-id*}

In this request, the URI variable *{metrics-context-id}* is the identifier of the metrics context object for which metrics are to be obtained.

Response body contents

On successful completion, the response body contains the set of metrics associated with the metrics context. The response is sent using HTTP chunked transfer encoding and UTF-8 character encoding. A MIME media type of **application/vnd.ibm-z-zmanager-metrics** is used and is specified in the **Content-Type** header on the response.

Because performance and scalability are a major concern for the metrics service, the response body is in a terse custom format, rather than being presented as a JSON object. The data type, name, and order information required to parse and interpret the response is provided in a previous **Create Metrics Context** response.

Data in this format will be delimited by newlines and commas.

Using a partial Extended Backus-Naur Form, where a comma (,) indicates concatenation and curly braces ({}) indicate 0 or more repetitions, we can express the format this way:

```
MetricsResponse = {MetricsGroup},NL

MetricsGroup = MetricsGroupName,NL,{ObjectValues},NL
MetricsGroupName = StringValue

NL = "\n"

ObjectValues = ObjectURI,NL,Timestamp,NL,ValueRows,NL

Timestamp = LongValue

ObjectURI = StringValue

ValueRows = ValueRow,{ValueRow}
ValueRow = Value,{",",Value},NL

Value = BooleanValue | ByteValue | DoubleValue | LongValue | IntegerValue | ShortValue | StringValue
```

The MetricsGroupName is the name of the metrics group, as a StringValue as defined below.

The Timestamp is the time when the associated values were buffered (i.e. “cached”) on the HMC. It is expressed as an “epoch” timestamp: a LongValue giving the milliseconds since January 1, 1970, 00:00:00 GMT (just as is expected, for example, by the constructor of a java.util.Date object).

The ObjectURI is the canonical URI of the object, as a StringValue as defined below.

NL is a single newline character (Unicode U+000A).

All the varieties of `Value` will be represented as strings according to the following rules and limits:

- BooleanValue
  - Either the string true or the string false.
- ByteValue
  - A string representation of a signed decimal integer in the range -128 to 127 (i.e. the range of a signed 8 bit integer).
- DoubleValue

  - A string representation of a 64 bit IEEE 754 floating point number in the range +/-4.9E-324 to +/-3.4028235E+38. Note that, although IEEE 754 provides for representations of positive or negative Infinity and NaN, such values are not allowed in the metric data feed and thus will not appear in a metrics service result. For results with a magnitude greater than or equal to 10^-3 and less than 10^7, the string representation will be a dotted decimal (e.g. 1.7, -32.467). For results with magnitudes outside that range, the string representation will be computerized scientific notation (e.g. -4.23E127).
- LongValue
  - A string representation of a signed decimal integer in the range -9223372036854775808 to 9223372036854775807 (i.e. the range of a signed 64 bit integer).
- IntegerValue
  - A string representation of a signed decimal integer in the range -2147483648 to 2147483647 (i.e. the range of a signed 32 bit integer).
- ShortValue
  - A string representation of a signed decimal integer in the range -32768 to 32767 (i.e. the range of a signed 16 bit integer).
- StringValue
  - A string starting with a double-quote, ending with a double-quote, and with any embedded double-quotes or backslashes escaped with a preceding backslash (i.e. escaped as \" and \\).

Description

On successful execution status code 200 (OK) is returned, with a response body as described above.

The URI path on the request must designate an existing metrics context for the current API session. If the URI designates an unrecognized context for the API session, then status code 404 (Not Found) is returned.

Note that under some circumstances the metrics service may delete a metrics context, requiring the client to establish a new context in order to resume metric retrievals. For example, the metrics service may choose to delete a given context if the time since the last associated **Get Metrics** request has exceeded 4 times the anticipated frequency specified when the context was created. In addition, the client may explicitly delete a metrics context with the **Delete Metrics Context** operation. If the URI designates a context that was once valid for the current API session, but no longer is, then status code 409 (Conflict) is returned.

Authorization requirements

Only metrics referencing managed objects accessible according to the permissions associated with the API session under which the Get Metrics is being issued will be returned. Note that there is no indication via an HTTP status or reason code that results may have been restricted due to authorization restrictions. Rather, success is indicated and the responses are just as if any restricted objects did not exist.

HTTP status and reason codes

On success, HTTP status code 200 (OK) is returned and the response body is provided as described in “Response body contents” on page 646.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See “Common request validation reason codes” on page 18 for a list of the possible reason codes.

HTTP error status code	Reason code	Description
404 (Not Found)	1	The metrics context ID in the URI (*{metrics-context-id}*) does not designate a metrics context for the associated API session.
490 (Conflict)	1	The metrics context ID in the URI (*{metrics-context-id}*) designates a metrics context for the associated API session that is no longer valid.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Usage notes

- Repeated metrics retrievals, even consecutive retrievals against a common metrics context, will not necessarily yield metrics for the exact same set of objects. Objects can come and go from the system's inventory due to various circumstances unrelated to the metrics data. The metrics feed for a particular metric group may stop for some reason, and the metrics data may therefore expire from the HMC's buffer (i.e. the metrics cache). The access permissions of a user associated with a metrics context may change, giving the user access to a smaller or larger set of objects (and therefore, perhaps, a smaller or larger set of metrics data).
- It is possible that there may be no metrics to return for one or more metric groups associated with the specified metrics context. For example, data for a metric group may not be buffered on the HMC at the time of the Get Metrics invocation, or authorization restrictions related to objects in a requested metric group may prevent any metrics for that group from being returned. If there is no metric data to be returned for a metric group name, then that group name does not appear in the response body. Furthermore, if there is no metric data to return for any metric group name associated with a context, the response body format above specifies that the body will consist only of a single newline character. This is nonetheless considered a successful response. In other words, an HTTP status code 200 (OK) will still be returned with such a response.

Example HTTP interaction

```
GET /api/services/metrics/context/1 HTTP/1.1
x-api-session: 4g33uc2vith3v42vmce8wjr5q7x58x5ybh6hwd4vjpwr7jl4sz
```

Figure 297. Get Metrics: Request

```
200 OK
transfer-encoding: chunked
server: zSeries management console API web server / 1.0
cache-control: no-cache, no-cache
date: Wed, 07 Dec 2011 04:38:20 GMT
content-type: application/vnd.ibm-z-zmanager-metrics;charset=UTF-8
"virtualization-host-cpu-memory-usage"
"/api/virtualization-hosts/2f7bf364-03f8-11e1-8eda-001f163805d8"
1323232689283
0,14,0,3,2942386000,4608,2942386000,0,0

"/api/virtualization-hosts/c14cde64-03e6-11e1-baf3-001f163805d8"
1323232689283
0,64,-1,-1,76028000,1311,-1,-1,-1

"/api/virtualization-hosts/5f805d28-03e6-11e1-baf3-001f163805d8"
1323232689283
3,72,-1,-1,1731046000,1485,-1,-1,-1

"/api/virtualization-hosts/634fa694-03f4-11e1-881f-001f163805d8"
1323232689283
4,36,-1,-1,55878116000,17920,-1,-1,-1
<3 blank lines here (consecutive new lines)>
```

Figure 298. Get Metrics: Response

Delete Metrics Context

The **Delete Metrics Context** operation deletes a metrics context.

HTTP method and URI

DELETE /api/services/metrics/context/{*metrics-context-id*}

In this request, the URI variable *{metrics-context-id}* is the identifier of the metrics context object for which metrics are to be obtained.

Description

This operation deletes the metrics context ID. That is, it disassociates it from the API session and cleans up any data associated with it. Further **Get Metrics** requests against this context will result in status code 409 (Conflict).

The URI path must designate an existing valid metrics context for the current API session. If the URI path represents an already invalidated metrics context for the current API session, status code 409 (Conflict) is returned. If the URI path does not represent a recognized metrics context for the current API session, status code 404 (Not Found) is returned.

Authorization requirements

There are no authorization requirements for deleting a metrics context. The association with the API session is implicit, so there is no possibility of deleting a context that was created by a different API session. In other words, only the session which created a metrics context can delete it.

HTTP status and reason codes

On success, HTTP status code 204 (No Content) is returned and no response body is provided.

The following HTTP status codes are returned for the indicated errors, and the response body is a standard error response body providing the reason code indicated and associated error message.

HTTP error status code	Reason code	Description
400 (Bad Request)	Various	Errors were detected during common request validation. See "Common request validation reason codes" on page 18 for a list of the possible reason codes.
404 (Not Found)	1	The metrics context ID in the URI (*{metrics-context-id}*) does not designate a metrics context for the associated API session.
490 (Conflict)	1	The metrics context ID in the URI (*{metrics-context-id}*) designates a metrics context for the associated API session that is no longer valid.

Additional standard status and reason codes can be returned, as described in Chapter 3, "Invoking API operations," on page 13.

Example HTTP interaction

```
DELETE /api/services/metrics/context/1 HTTP/1.1
x-api-session: 6a9oz3ymut6rvjijrft0loqhfzgpp0rnu4mjishwh6d39jh31q
```

Figure 299. Delete Metrics Context: Request

```
204 No Content
date: Wed, 07 Dec 2011 04:01:59 GMT
server: zSeries management console API web server / 1.0
cache-control: no-cache, no-cache

<No response body>
```

Figure 300. Delete Metrics Context: Response

Chapter 16. zManager metric groups

This chapter provides a description of the metric groups that can be retrieved using the Metrics Service. For each metric group provided by the HMC, the material in this chapter describes the purpose and general characteristics of the metric group, and then defines the content of the metric group via a table that specifies the metric fields provided by the group. The order in which metric fields appear within these tables corresponds to the order in which the data items appear in a value row returned by the **Get Metrics** operation. For example, the metric field appearing in the first row of a metric group table (and identified in a table below as being in position 1) will be the first data item provided in a value row for that metric group; the metric field appearing in the second row (position 2) will be the next data item in a value row, and so on. Thus, the order in which metric fields are documented here is considered semantically significant and can be relied upon by client applications in order to simplify parsing of the data retrieved using the **Get Metrics** operation.

The contents of metric groups may be extended in future versions of this API. If a metric group is extended, new metric fields will be added to the end so as to not alter the relative positions of any of the existing fields. Such new fields would not be understood by a client application designed for an earlier version of the API. Therefore, applications must be developed using the philosophy of "ignore what you don't understand/expect" when processing metric group data order to tolerate such possible future extensions. See “Compatibility” on page 4 for more discussion on API compatibility principles.

Monitors dashboard metric groups

The Monitors Dashboard task is the current system monitoring interface on the HMC. It gives a dashboard display to monitor system resources, such as power consumption, environmental data, processor usage, etc.

In order to provide programmatic access to this same data, the utilization and environment data that is displayed on the user interface is also provided via the Metrics Service in the following metric groups.

BladeCenter temperature and power metric group

This metric group reports the ambient temperature and power consumption for each BladeCenter on the system.

Metric Group Name
"bladecenter-temperature-and-power"

Collection Interval
15 seconds

Applicable Managed Object Class
"bladecenter"

The following metrics are provided in each entry of this metric group:

Table 151. BladeCenter temperature and power metric group

Pos	Metric field name	Type	Units	Description
1	**temperature-celsius**	Double	Degrees Celsius	The ambient temperature
2	**power-consumption-watts**	Integer	Watts	The power consumption

Blade power

This metric group reports power consumption for each blade on the system.

Metric Group Name
"blade-power-consumption"

Collection Interval
15 seconds

Applicable Managed Object Class
"blade"

The following metrics are provided in each entry of this metric group:

Table 152. Blade power metric group

Pos	Metric field name	Type	Units	Description
1	**power-consumption-watts**	Integer	Watts	The power consumption

Channels

This metric group reports the channel usage for each channel on the system. An instance of this metric group is created for each channel of a CPC.

Metric Group Name
"channel-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 153. Channels metric group

Pos	Metric field name	Type	Units	Description
1	**channel-name**	String	—	The name of the channel in the form CSS.Chpid
2	**shared-channel**	Boolean	—	True if the channel is shared among logical partitions, and false if it is not
3	**logical-partition-name**	String	—	For channel types for which logical partition names are available, the name of the owning logical partition or the value **"shared"** if the channel is shared. For other channel types for which the name is not available (for example, coupling channels), the value is always an empty string.
4	**channel-usage**	Integer	%	The channel percent usage (0 – 100%)

CPC overview

This metric group reports the aggregated processor usage and channel usage, the ambient temperature, and total system power consumption for each system. The cpc-processor-usage is the average of the percentages of processing capacity for the physical processor lines in the active System Activity profile for the CPC. The channel-usage is the average of the percentages of I/O capacity for the channel lines in the active System Activity profile for the CPC.

Metric Group Name
"cpc-usage-overview"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 154. CPC overview metric group

Pos	Metric field name	Type	Units	Description
1	**cpc-processor-usage**	Integer	%	The processor percent usage.
2	**channel-usage**	Integer	%	The channel percent usage.
3	**power-consumption-watts**	Integer	Watts	The total system power consumption. This includes CPC, BladeCenters, and blades.
4	**temperature-celsius**	Double	Degrees Celsius	The ambient temperature.
5	**cp-shared-processor-usage**	Integer	%	The processor percent usage for all CP shared processors. Set to -1 if there are no processors of this type.
6	**cp-dedicated-processor-usage**	Integer	%	The processor percent usage for all CP dedicated processors. Set to -1 if there are no processors of this type.
7	**ifl-shared-processor-usage**	Integer	%	The processor percent usage for all IFL shared processors. Set to -1 if there are no processors of this type.
8	**ifl-dedicated-processor-usage**	Integer	%	The processor percent usage for all IFL dedicated processors. Set to -1 if there are no processors of this type.
9	**icf-shared-processor-usage**	Integer	%	The processor percent usage for all ICF shared processors. Set to -1 if there are no processors of this type.
10	**icf-dedicated-processor-usage**	Integer	%	The processor percent usage for all ICF dedicated processors. Set to -1 if there are no processors of this type.
11	**iip-shared-processor-usage**	Integer	%	The processor percent usage for all zIIP shared processors. Set to -1 if there are no processors of this type.
12	**iip-dedicated-processor-usage**	Integer	%	The processor percent usage for all zIIP dedicated processors. Set to -1 if there are no processors of this type.
13	**aap-shared-processor-usage**	Integer	%	The processor percent usage for all zAAP shared processors. Set to -1 if there are no processors of this type.
14	**aap-dedicated-processor-usage**	Integer	%	The processor percent usage for all zAAP dedicated processors. Set to -1 if there are no processors of this type.
15	**all-shared-processor-usage**	Integer	%	The processor percent usage for all shared processors. Set to -1 if there are no processors of this type.
16	**all-dedicated-processor-usage**	Integer	%	The processor percent usage for all dedicated processors. Set to -1 if there are no processors of this type.

Logical partitions

This metric group reports the processor usage and z/VM paging rate for each active logical partition (aka Image, LPAR Image, Zone, PR/SM virtual server) on the system.

Metric Group Name
"logical-partition-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"logical-partition"

The following metrics are provided in each entry of this metric group:

Table 155. Logical partitions metric group

Pos	Metric field name	Type	Units	Description
1	**processor-usage**	Integer	%	The processor percent usage.
2	**zvm-paging-rate**	Integer	Pages Per Second	The z/VM paging rate. Only returned for logical partitions running z/VM level 6.1 or higher that have the appropriate agent running in them.

zCPC environmentals and power

This metric group reports environmental data and power consumption for the zCPC, which is the legacy part of the system; i.e. without blades.

Metric Group Name
"zcpc-environmentals-and-power"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 156. zCPC environmentals and power metric group

Pos	Metric field name	Type	Units	Description
1	**temperature-celsius**	Double	Degrees Celsius	The ambient temperature
2	**humidity**	Integer	%	The relative humidity
3	**dew-point-celsius**	Double	Degrees Celsius	The dew point
4	**power-consumption-watts**	Integer	Watts	The power consumption in watts

zCPC processors

This metric group reports the processor usage for each physical zCPC processor on the system. This includes the System Assist Processors (SAPs) and does not include blades. An instance of this metric group is created for each processor of a CPC.

Metric Group Name
"zcpc-processor-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 157. zCPC processors metric group

Pos	Metric field name	Type	Units	Description
1	**processor-name**	String		The name of the zcpc processor in the form processor-type + processor ID. For example, IFL01.
2	**processor-type**	String		The type of zcpc processor. The valid types are: **"cp"**, **"ifl"**, **"icf"**, **"iip"**, **"aap"**, **"sap"**.
3	**processor-usage**	Integer	%	The processor percent usage.

Blade CPU and memory metric group

This metric group reports CPU and memory utilization for each of the blades in the ensemble. This group provides data for all types of blades.

Metric Group Name
"blade-cpu-memory-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"blade"

The following metrics are provided in each entry of this metric group:

Table 158. Blade CPU and memory metric group

Pos	Metric field name	Type	Units	Description
1	**processor-usage**	Integer	%	Processor utilization percentage (0-100%). Value for current interval. If the value is not currently available, for example if the blade is powered off, -1 is provided.
2	**memory-usage**	Integer	%	Memory utilization percentage (0-100%). Value for current interval. If the value is not currently available, for example if the blade is powered off, -1 is provided.

Cryptos

This metric group reports the adapter usage for each crypto on the system. An instance of this metric group is created for each crypto of a CPC. If a CPC has no crypto adapters, then no data will appear in this metric group for that CPC.

Metric Group Name
"crypto-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 159. Crypto metric group

Pos	Metric field name	Type	Units	Description
1	**channel-id**	String		The physical channel identifier of the crypto
2	**crypto-id**	String		The crypto identifier of the crypto, hex char 0-F
3	**adapter-usage**	Integer	%	The adapter percent usage (0-100%)

Flash Memory Adapters

This metric group reports the adapter usage for each Flash memory (Flash Express) adapter on the system. An instance of this metric group is created for each Flash memory adapter of the CPC. If a CPC has no flash memory adapters, then no data will appear in this metric group for that CPC.

Note: Flash Express has a planned exploitation of December 2012.

Metric Group Name
"flash-memory-usage"

Collection Interval
15 seconds

Applicable Managed Object Class
"cpc"

The following metrics are provided in each entry of this metric group:

Table 160. Flash memory adapters metric group

Pos	Metric field name	Type	Units	Description
1	**channel-id**	String		The physical channel identifier of the Flash memory adapter
2	**adapter-usage**	Integer	%	The adapter percent usage (0-100%)

Performance management metrics groups

Following are the performance management metrics groups.

Virtual server CPU and memory metrics group

This metric group is collected for all types of virtual servers: PowerVM, PR/SM, x Hyp, and z/VM.

Metric Group Name
"virtual-server-cpu-memory-usage"

Collection Interval
15 seconds for PR/SM, x Hyp and z/VM virtual servers

30 seconds for PowerVM virtual servers

Applicable Managed Object Class
"virtual-server"

The following metrics are provided in each entry of this metric group:

Table 161. Virtual server CPU and memory metric group

Pos	Metric field name	Type	Units	Description
1	**processor-usage**	Integer	%	Physical utilization percentage (0-100%). Values for current interval.
2	**memory-usage**	Integer	%	Memory usage percentage (0-100%). Value for current interval.
3	**up-time**	Long	Microseconds	Time since the virtual server was started. Not supported for PR/SM, will be reported as -1.
4	**physical-cpu-time**	Long	Microseconds	Physical CPU time used by virtual server. This time accumulates until the virtualization host or the support element is restarted.

Table 161. Virtual server CPU and memory metric group (continued)

Pos	Metric field name	Type	Units	Description
5	**virt-host-cpu-delay-time**	Long	Microseconds	The virtual processors were ready to run but not dispatched on physical processors due to competition with other virtual servers. This time accumulates until the virtualization host or the support element is restarted.Not supported for PR/SM, will be reported as -1.
6	**elapsed-time**	Long	Microseconds	Elapsed time over which physical-cpu-time and virt-host-cpu-delay-time have accumulated.
7	**cpu-allocation**	Integer	Share for z/VM, and physical CPU equivalent for PowerVM, and processing weight for PR/SM	Amount of CPU resource allocated to virtual server. zManager adjusts this value when virtual server management is enabled. For PowerVM, this is the processing units setting. For z/VM this is the CPU share value. For PS/SM this is the general purpose processing weight setting. Not supported for x Hyp or virtual servers with dedicated virtual processors, will be reported as -1.
8	**current-physical-memory-used**	Integer	Megabytes	Amount of physical memory currently used by virtual server.
9	**os-total-cpu-using-samples**[1]	Long	Count	Count of samples where virtual CPUs were seen in use. Will be 0 if GPMP not running on virtual server.
10	**os-total-cpu-delay-samples**[1]	Long	Count	Count of samples where threads were delayed waiting for virtual CPUs. Will be 0 if GPMP not running on virtual server.
11	**os-total-paging-delay-samples**[1]	Long	Count	Count of samples where threads were delayed waiting for page faults to be resolved. Will be 0 if GPMP not running on virtual server.
12	**os-total-io-delay-sample**[1]	Long	Count	Count of samples where threads were delayed waiting for I/O to complete. Will be 0 if GPMP not running on virtual server.
13	**os-sampling-rate**	Integer	Count	Number of times per second OS samples are collected. Will be 0 if GPMP not running on virtual server.
14	**os-total-other-samples**[1]	Long	Count	Count of samples where processes where in a state not included in the other sample counts.
15	**gpmp-active**	Boolean		True if the GPMP was active on the virtual server at the end of the interval. False otherwise.
16	**other-time**	Long	Microseconds	Sum of time any of the virtual processors of the virtual server where in a state other than dispatched on physical processors, delayed, or idle.
17	**idle-time**	Long	Microseconds	Sum of the time any of the virtual processors of the virtual server were idle.
18	**virtual-cpu-count**	Integer	Count	Number of virtual processors defined for the virtual server, totaled across all types (dedicated/shared, CP/IFL/zIIP/zAAP)

Table 161. Virtual server CPU and memory metric group (continued)

Pos	Metric field name	Type	Units	Description
	Note: 1. On an interval basis the GPMP samples the state of OS threads. Each sampling interval the GPMP counts the number of threads in various states. Each of these states represents a sample type. The sample types are: • CPU Using: Each sample represents a thread found actively running on a virtual processor • CPU delay: Each sample represents a thread waiting to be dispatched on a virtual processor • Page Delay: Each sample represents a thread waiting for a page fault to be resolved • I/O Delay: Each sample represents a threads waiting for I/O to complete • Other: Each sample represents processes that had no threads in one of the above states. The GPMP keeps a running total for each sample type. These running totals are returned in the sample metrics that are part of the virtual-server-cpu-memory-usage metrics group. The sampling interval is returned in the os-sampling-rate metric.			

Virtualization host CPU and memory metrics group

Metric Group Name
"virtualization-host-cpu-memory-usage"

Collection Interval
15 seconds for PR/SM, x Hyp and z/VM virtualization hosts

30 seconds for PowerVM virtualization hosts

Applicable Managed Object Class
"virtualization-host"

The following metrics are provided in each entry of this metric group:

Table 162. Virtualization host CPU and memory metric group

Pos	Metric field name	Type	Units	Description
1	**processor-usage**	Integer	%	Overall CPU utilization percentage for the virtualization host (0-100%). Value for current interval. For PR/SM only includes general purpose processors.
2	**memory-usage**	Integer	%	Memory usage percentage for the virtualization host (0-100%). Value for current interval.
3	**network-usage**	Integer	%	Network utilization percentage for the virtualization host (0-100%). Value for current interval. Not available for z/VM or PR/SM, will be reported as -1.
4	**storage-rate**	Integer	Kbytes per sec	Average Kbytes transferred to disk over interval. Value for current interval. Not available for z/VM or PR/SM, will be reported as -1.
5	**physical-cpu-time**	Long	Microseconds	Physical CPU time used. Cumulative value.For PR/SM only includes general purpose processors.
6	**memory-used**	Integer	Mbytes	Current memory in use by the virtualization host.

Table 162. Virtualization host CPU and memory metric group (continued)

Pos	Metric field name	Type	Units	Description
7	**virt-host-management-cpu-time-used**	Long	Microseconds	CPU time used for virtualization host management. Cumulative value. Note for PowerVM this is the CPU used by the VIOS partition. For x Hyp this is the Linux system mode CPU time. Currently not supported for z/VM or PR/SM, will be reported as -1.
8	**virt-host-page-ins**	Long	Count	Paging activity by the virtualization host to support hypervisor management. Page reads from paging space. Cumulative value. For PowerVM this represents VIOS paging. For x Hyp it is the base Linux paging. Currently not supported for z/VM or PR/SM, will be reported as -1.
9	**virt-host-page-outs**	Long	Count	Paging activity by the virtualization host to support hypervisor management. Page written to paging space. Cumulative value. For PowerVM this represents VIOS paging. For x Hyp it is the base Linux paging. Currently not supported for z/VM or PR/SM, will be reported as -1.
10	**cp-cpu-time**[1]	Long	Microsecond	CPU time accumulated by the general purpose processors owned by the virtualization host. Only supported for z/VM, will be reported as -1 for other hypervisor types.
11	**ifl-cpu-time**[1]	Long	Microsecond	CPU time accumulated by the IFL processors owned by the virtualization host. Only supported for z/VM, will be reported as -1 for other virtualization host types.
12	**zaap-cpu-time**[1]	Long	Microsecond	CPU time accumulated by the zAAP processors owned by the virtualization host. Only supported for z/VM, will be reported as -1 for other virtualization host types.
13	**ziip-cpu-time**[1]	Long	Microsecond	CPU time accumulated by the zIIP processors owned by the virtualization host. Only supported for z/VM, will be reported as -1 for other virtualization host types.
14	**icf-cpu-time**[1]	Long	Microsecond	CPU time accumulated by the ICF processors owned by the virtualization host. Only supported for z/VM, will be reported as -1 for other virtualization host types.

Table Notes:

1. Provided for virtualization hosts on CPCs with SE version 2.12.0 or later; metric not present for CPCs with earlier SE versions.

Workload service class data metrics group

This metric group reports workload performance data on a per-service-class basis. At each collection interval for a given workload, one instance of this metric group is added to the metric cache for each service class of the active policy for that workload.

Metric Group Name
"workload-service-class"

Collection Interval
15 seconds

Applicable Managed Object Class
"workload"

The following metrics are provided in each entry of this metric group:

Table 163. Workload metrics group - service class data metric group

Pos	Metric field name	Type	Units	Description
1	**policy-activation-time**	Long	Time stamp	Time of the last policy activation for this workload. This is the last-activation-requested-date property of the currently active policy.
2	**service-class-name**	String (1-64)	—	Name of service class
3	**velocity-numerator**	Long	Microseconds	Time value used for numerator of velocity calculation. Cumulative value since last policy activation.
4	**velocity-denominator**	Long	Microseconds	Time value used for denominator of velocity calculation. Cumulative value since last policy activation.

Network management metrics

Following are the network management metric groups.

Virtualization host and virtual server metrics

The zManager Metrics Service provides network metrics for zEnterprise network resources. These metrics are collected at the virtualization host (hypervisor) network layer. The virtualization host provides a virtual LAN for network communications between the virtual servers it is hosting, and transparently virtualizes the attached physical network interfaces providing server access to the physical network. In many cases, this network virtualization function within the virtualization host is commonly referred to as the “virtual switch” or “vSwitch”. In zManager, although only certain zManager platforms allow for explicit external exposure and management of the vSwitch, the network metrics collected at this layer will be exposed for each platform. In some cases, such as **"zvm"** and **"prsm"**, the virtualization host supports multiple vSwitches, and the vSwitches are externally identified within zManager. For example, in zManager, OSX and IQD-X chpids are referred to as vSwitches. The term “vSwitch” is also used to generically describe the network functions of the virtualization host. For **"x-hyp"**, **"power-vm"**, and **"prsm"** virtualization host types, a specific vSwitch resource is not explicitly externalized for management within zManager; therefore, it is the virtualization host itself that is associated with the metrics, and, where the metric group provides a vSwitch name, this value will be “N/A”. In these cases, the virtualization host implicitly represents a single vSwitch.

The network metrics provided at the virtualization host are the following:

- Virtualization host uplink metrics: Virtual network (i.e. per VLAN ID) metrics are not provided for uplinks. . Metrics collected are provided on an interval. These metrics can be collected from the Virtualization Host (vSwitch) Uplink Metric Group. Metrics for two types of uplinks are provide:
  - Real Uplinks: These are metrics which are captured between the virtualization host or virtualization host's vSwitch and the attached physical network interfaces. These metrics can show the bandwidth that the virtualization host is contributing to the IEDN.
  - Virtual Uplinks: These metrics are captured by these vSwitch ports which are not directly attached to a physical vSwitch that attaches to the IEDN. The following describe the supported virtual uplinks:
    - An IQD-X chpid that is attached to a z/VM LPAR's vSwitch bridge port. The z/VM vSwitch bridge port provides the uplink from the IQD-X vSwitch.

  - A z/VM vSwitch virtual uplink. A virtual uplink connects a vSwitch to a z/VM virtual server. In this case, traffic is forwarded to this server, which may be useful for packet collection for debugging or analysis.
- Virtualization host by vSwitch by virtual network metrics: These metrics are captured between the virtualization host and its virtual servers. In cases, such as z/VM, where a virtualization host has multiple vSwitches, the metrics are captured by virtualization host by vSwitch by virtual network. These metrics can be collected from the Virtualization Host (vSwitch) by Virtual Network Metric Group.
- Network metrics from virtualization host for each attached virtual server's virtual network adapter by virtual network (VLAN ID): These metrics can be collected from the Attached Virtual Servers Network Adapter Metric Group.

In general, for the Virtualization Host by Virtual Network Metrics Group and the Attached Virtual Servers Network Adapter Metric Group, the metrics are collected at the virtualization host level for the attached virtual server virtual network adapter by virtual network (VLAN ID). The network metrics collected at this level provide a view of the performance between the virtual switch and the virtual server, with metrics such as bytes sent and bytes received. Other metrics such as packets dropped or discarded can help to determine if problems are occurring. Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Providing metrics at the virtualization host's virtual network adapters provides a level of granularity that allows for the consumer to aggregate these metrics. Use of these metrics along with configuration data provided through the zManager external API allow the client application to determine resource utilization relationships.

Virtualization host (vSwitch) uplink metric group

This metric group provides virtualization host (hypervisor)-based network metrics for the uplink ports. The metrics provided by this group represent the uplink metrics between the vSwitch and the physical network interface. These metrics are from the perspective of the vSwitch sending to and receiving from the physical network interfaces.

In the case of z/VM, there may be multiple uplink interfaces; therefore, multiple instances of this metric group may be provided for a single z/VM virtualization host. This is also true for a PR/SM virtualization host. For a PR/SM virtualization host, there may be multiple OSX's for which the uplink information provides the metrics between the OSX and the physical network.

Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Metric Group Name
"network-virtualization-host-uplink"

Collection Interval
30 seconds

Applicable Managed Object Class
"virtualization-host"

The following metrics are provided in each entry of this metric group:

Table 164. Virtualization host (vSwitch) uplink metric group

Pos	Metric field name	Type	Units	Description
1	**uplink-id**	String		Name of the uplink interface. The uplink names are described as follows and the naming convention will be unique based upon the **type** property of the virtualization-host object: • **"power-vm"** - The names will be the platform-defined names of the physical network interfaces, for example "ent*x*" or "ent*y*". Where *x* and *y* are numeric values. • **"x-hyp"** - The names will be the platform-defined names of the physical network interfaces, for example "ent*x*" or "ent*y*". Where *x* and *y* are numeric values. • **"prsm"** - where the uplink is an OSX, the uplink is the pchid. The name will be in the format of "OSX *pchid.port*" • **"prsm"** - where the uplink is an IQDX chpid that is connected to a z/VM vSwitch bridge port. The name will be in the form "IQDX *css.chpid:zvm lpar name.vswitch name*." • **"zvm"** - If the uplink is OSX, then the uplink will identify the OSA css and chpid, the z/VM LPAR name and virtual device number of the OSX. The name will be in the format of "OSX css.*chpid:zvm lpar name.vdev number*".
2	**uplink-type**	String Enum		The uplink type is: • **"real"** • **"virtual"**
3	**vswitch-name**	String		Name of the vSwitch. In the case where the vSwitch is not uniquely identified for a virtual host then this will be "N/A". "N/A" is returned for all virtualization-host objects except where the **type** property of the virtualization-host object is **"zvm"**.
4	**bytes-sent**	Long	Count	Number of bytes sent from the uplink interface to the physical network.
5	**bytes-received**	Long	Count	Number of bytes received by the uplink interface from the physical network.
6	**packets-sent**	Long	Count	Number of packets sent from the uplink interface to the physical network.
7	**packets-received**	Long	Count	Number of packets received by this uplink interface from the physical network.
8	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending from this uplink interface to the physical network. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.
9	**packets-received-dropped**	Long	Count	Number of packets received by this uplink interface from the physical network that were dropped. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.

Table 164. Virtualization host (vSwitch) uplink metric group (continued)

Pos	Metric field name	Type	Units	Description
10	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from the uplink interface to the physical network. Packets may be discarded due to errors associated with the packet, such as malformed packets.
11	**packets-received-discarded**	Long	Count	Number of packets received by this uplink interface that were discarded. Packets may be discarded due to errors associated with the packet, such as malformed packets.
12	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from the uplink interface to the physical network.
13	**multicast-packets-received**	Long	Count	Number of multicast packets received by the uplink interface from the physical network.
14	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from the uplink interface to the physical network.
15	**broadcast-packets-received**	Long	Count	Number of broadcast packets received by this uplink interface from the physical network.
16	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this uplink interface to the physical network over the collection interval.
17	**interval-bytes-received**	Long	Bytes	Number of bytes received by this uplink interface from the physical network over the collection interval.
18	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this uplink interface to the physical network over the collection interval.
19	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received per second by this uplink interface from the physical network over the collection interval.
20	**mac-address**	String		The MAC address of this uplink, if known. If it is not known then "N/A".
21	**flags**	Long		Flags indicating the types of metrics that are reported by this uplink. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Virtualization host (vSwitch) by virtual network metric group

The virtualization host (vSwitch) Virtual Network metric collection group provides virtual network metrics by virtualization host by vSwitch by virtual network. These metrics are collected at the virtualization host's vSwitch port level by virtual network and aggregated to provide metrics for each vSwitch by virtual network. For each virtualization host there will be an instance of these metrics for each vSwitch (in cases where multiple vSwitches are associated with the virtualization host) for each virtual network (vlan-id); therefore, there may be multiple instances within the group. In cases where a "vSwitch" is not externalized for the virtualization host, the name of the vSwitch is "N/A" and the metrics are associated with the virtualization host. These metrics are essentially an aggregation of the metrics from the "Attached virtual server network adapters metric group" on page 665.

Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Metric Group Name
"network-vswitch-by-virtual-network"

Collection Interval
30 seconds

Applicable Managed Object Class
"virtualization-host"

The following metrics are provided in each entry of this metric group:

Table 165. Virtualization host (vSwitch) by virtual network metric group

Pos	Metric field name	Type	Units	Description
1	**vswitch-name**	String		Name of the vSwitch. In the case where the vSwitch is not uniquely identified for a virtual host then this will be “N/A”. “N/A” is returned for all virtualization-host objects except where the **type** property of the virtualization-host object is **"zvm"**.
2	**vlan-id**	Integer		VLAN ID of the virtual network for which metrics are being provided. This value corresponds to the **vlan-id** property of the related virtual network object. There may be cases where this field is 0 when metrics are reported and the VLAN ID is unable to be determined.
3	**bytes-sent**	Long	Count	Number of bytes sent from this virtualization host or virtualization host's vSwitch to the attached virtual servers.
4	**bytes-received**	Long	Count	Number of bytes received by this virtualization host or virtualization host's vSwitch from the attached virtual servers.
5	**packets-sent**	Long	Count	Number of packets sent from this virtualization host or virtualization host's vSwitch to the attached virtual servers.
6	**packets-received**	Long	Count	Number of packets received by this virtualization host or virtualization host's vSwitch from the attached virtual servers.
7	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending from this virtualization host or virtualization host's vSwitch to the attached virtual servers. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.
8	**packets-received-dropped**	Long	Count	Number of packets received by this virtualization host or virtualization host's vSwitch from the attached virtual servers that were dropped. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.

Table 165. Virtualization host (vSwitch) by virtual network metric group (continued)

Pos	Metric field name	Type	Units	Description
9	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from this virtualization host or virtualization host's vSwitch to the attached virtual servers. Packets may be discarded due to errors associated with the packet, such as malformed packets.
10	**packets-received-discarded**	Long	Count	Number of packets received by this virtualization host or virtualization host's vSwitch from the virtual servers that were discarded. Packets may be discarded due to errors associated with the packet, such as malformed packets
11	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from this virtualization host or virtualization host's vSwitch to the attached virtual servers.
12	**multicast-packets-received**	Long	Count	Number of multicast packets received by this virtualization host or virtualization host's vSwitch from the attached virtual servers.
13	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from this virtualization host or virtualization host's vSwitch to the attached virtual servers.
14	**broadcast-packets-received**	Long	Count	Number of broadcast packets received by this virtualization host or virtualization host's vSwitch from the attached virtual servers.
15	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this virtual switch, for the VLAN specified by **vlan-id**, over the collection interval.
16	**interval-bytes-received**	Long	Bytes	Number of bytes received by this virtual switch, for the VLAN specified by **vlan-id**, over the collection interval.
17	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this virtual switch, for the VLAN specified by **vlan-id**, over the collection interval.
18	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received by this virtual switch, for the VLAN specified by **vlan-id**, over the collection interval.
19	**flags**	Long		Flags indicating the types of metrics that are supported. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Attached virtual server network adapters metric group

This metric group provides metrics for each virtual server's virtual network adapter by virtual network. Each virtual network adapter may be associated with multiple virtual networks; therefore, there may be multiple instances of these metrics within the group. These metrics are collected at the virtualization host's (vSwitch) port level for the attached virtual server virtual network adapter by virtual network. These metrics are provided from the perspective of the virtualization host or virtualization host's vSwitch sending data to and receiving data from the virtual server's network adapter. However, there is an exception to this in the case of the PowerVM virtualization host. In this case, the metrics are collected at

the virtual server's guest O/S level. The metrics are included in this list for consistency, allowing for a good approximation of the data flowing over the interfaces. Therefore, the metric counters are reversed from what is provided for the metric in this group. Counters like drops and discards are from the perspective of the guest O/S for this interface and not the virtualization host.

The resource ID of the associated virtual server's network adapter is provided to allow the client application to correlate the metrics with a particular virtual server.

Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Metric Group Name
"network-virtual-server-attached-network-adapter"

Collection Interval
30 seconds

Applicable Managed Object Class
"virtual-server"

The following metrics are provided in each entry of this metric group:

Table 166. Attached virtual server network adapters metric group

Pos	Metric field name	Type	Units	Description
1	**network-adapter-id**	String		Element identifier of the virtual servers' network adapter. This value corresponds to the *{network-adapter-id}* portion of the URI for the network adapter. The full URI can be constructed using this value together with the URI of the parent virtual server.
2	**vlan-id**	Integer		VLAN ID of the virtual network for which metrics are being provided. This value corresponds to the **vlan-id** property of the related virtual network object. There may be cases where this field is 0 when zManager is unable to determine the per virtual network metrics for this interface.
3	**mac-address**	String		MAC address of this interface.
4	**bytes-sent**	Long	Bytes	Number of bytes sent to this Virtual Server network interface for the virtual network.
5	**bytes-received**	Long	Bytes	Number of bytes received from this Virtual Server network interface for the virtual network.
6	**packets-sent**	Long	Count	Number of packets sent on this interface to the Virtual Server for the virtual network.
7	**packets-received**	Long	Count	Number of packets received from this Virtual Server network interface for the virtual network.
8	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending to this Virtual Server Network interface for the virtual network. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.

Table 166. Attached virtual server network adapters metric group (continued)

Pos	Metric field name	Type	Units	Description
9	**packets-received-dropped**	Long	Count	Number of packets received from this Virtual Server network interface and for this virtual network that were dropped. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.
10	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending to this Virtual Server network interface for this virtual network. Packets may be discarded due to errors associated with the packet, such as malformed packets.
11	**packets-received-discarded**	Long	Count	Number of packets received from this virtual server network interface for this virtual network that were discarded. Packets may be discarded due to errors associated with the packet, such as malformed packets
12	**multicast-packets-sent**	Long	Count	Number of multicast packets sent to this virtual server network interface for this virtual network.
13	**multicast-packets-received**	Long	Count	Number of multicast packets received from this virtual server network interface for this virtual network.
14	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent to this virtual server virtual network interface for this virtual network.
15	**broadcast-packets-received**	Long	Count	Number of broadcast packets received from this virtual server virtual network interface for this virtual network.
16	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this network adapter over the collection interval.
17	**interval-bytes-received**	Long	Bytes	Number of bytes received by this network adapter over the collection interval.
18	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this network adapter over the collection interval.
19	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received per second by this network adapter over the collection interval.
20	**flags**	Long		Flags indicating the types of metrics that are supported by this interface. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Optimizer network metrics

Network metrics are provided for optimizer blades. The following metric groups are provided:

- Optimizer IEDN Virtual Network Interface Metric Group
- Optimizer IEDN Physical Network Adapter Metric Group

Optimizer IEDN virtual network interface metric group

This metric group provides metrics for an optimizer's IEDN attached network interfaces by virtual network. Currently the following optimizers are supported for this metric group: Datapower.

Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Metric Group Name
"network-optimizer-attached-iedn-interface"

Collection Interval
30 seconds

Applicable Managed Object Class
"blade"

The following metrics are provided in each entry of this metric group:

Table 167. Optimizer IEDN virtual network interface metric group

Pos	Metric field name	Type	Units	Description
1	**iedn-interface-id**	String		Element identifier of the optimizer's IEDN interface. This value corresponds to the *{iedn-interface-id}* portion of the URI for the interface. The full URI can be constructed using this value together with the URI of the parent blade.
2	**vlan-id**	Integer		VLAN ID of the virtual network for which metrics are being provided. This value corresponds to the **vlan-id** property of the related virtual network object. There may be cases where this field is 0 when zManager is unable to determine the per virtual network metrics for this interface.
3	**mac-address**	String		MAC address of this interface.
4	**bytes-sent**	Long	Count	Number of bytes sent from this interface.
5	**bytes-received**	Long	Count	Number of bytes received by this interface.
6	**packets-sent**	Long	Count	Number of packets sent from this interface.
7	**packets-received**	Long	Count	Number of packets received by this interface.
8	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending to this Virtual Server Network interface for the virtual network. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.
9	**packets-received-dropped**	Long	Count	Number of packets received by this interface that were dropped. Packets may be dropped due to conditions related to resource constraints such as a buffer shortage.
10	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from this interface. Packets may be discarded due to errors associated with the packet, such as malformed packets.

Table 167. Optimizer IEDN virtual network interface metric group (continued)

Pos	Metric field name	Type	Units	Description
11	**packets-received-discarded**	Long	Count	Number of packets received by this interface that were discarded. Packets may be discarded due to errors associated with the packet, such as malformed packets.
12	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from this interface.
13	**multicast-packets-received**	Long	Count	Number of multicast packets received by this interface
14	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from this interface.
15	**broadcast-packets-received**	Long	Count	Number of broadcast packets received to this interface.
16	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this interface over the collection interval.
17	**interval-bytes-received**	Long	Bytes	Number of bytes received by this interface over the collection interval
18	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this interface over the collection interval.
19	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received per second by this interface over the collection interval.
20	**flags**	Long		Flags indicating each types of metrics are supported. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Optimizer IEDN physical network adapter metric group

This metric group provides metrics for an optimizer's IEDN attached network interfaces. Currently the following optimizers are supported for this metric group: Datapower.

Metrics are collected and provided on an interval, and each metric provided is the total cumulative value, and not a delta.

Metric Group Name
"optimizer-physical-network-adapter"

Collection Interval
30 seconds

Applicable Managed Object Class
"blade"

This metric collection provides metrics for an optimizer's physical network adapters.

Table 168. Optimizer IEDN physical network adapter metric group

Pos	Metric field name	Type	Units	Description
1	**network-adapter-id**	String		The network-adapter-id is the name of the optimizer's physical network adapter. For the DataPower optimizer, there are two physical network interfaces. For example, names are commonly **"eth7"** and **"eth9"**.
2	**mac-address**	String		MAC address of this interface
3	**bytes-sent**	Long	Count	Number of bytes sent from this interface.
4	**bytes-received**	Long	Count	Number of bytes received by this interface
5	**packets-sent**	Long	Count	Number of packets sent from this interface
6	**packets-received**	Long	Count	Number of packets received by this interface.
7	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending from this interface.
8	**packets-received-dropped**	Long	Count	Number of packets received by this interface that were dropped.
9	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from this interface. Packets may be discarded due to errors associated with the packet, such as malformed packets.
10	**packets-received-discarded**	Long	Count	Number of packets received by this interface that were discarded. Packets may be discarded by the virtual or physical due to errors associated with the packet, such as malformed packets.
11	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from this interface.
12	**multicast-packets-received**	Long	Count	Number of multicast packets received by this interface.
13	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from this interface.
14	**broadcast-packets-received**	Long	Count	Number of broadcast packets received by this interface.
15	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this network adapter over the collection interval.
16	**interval-bytes-received**	Long	Bytes	Number of bytes received by this network adapter over the collection interval.
17	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes received per second by this network adapter over the collection interval.
18	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes sent per second by this network adapter over the collection interval.

Table 168. Optimizer IEDN physical network adapter metric group (continued)

Pos	Metric field name	Type	Units	Description
19	**flags**	Long		Flags indicating the types of metrics that are reported by this uplink. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Physical switches

The physical switches provide the connectivity between the blades and CPCs in the zEnterprise intraensemble data network (IEDN).

There are two types of Ethernet switches within each zBX:
- Top-of-Rack Switches (TORs) – A pair of TORs reside in each zBX and act as a primary and backup. TORs connect the blades in the zBX to System z network interfaces, and to other external networking equipment, such as routers.
- Ethernet Switch Modules (ESMs) - These switches connect the blades in the zBX to the IEDN and link the blades to the TORs.

The initial configuration and setup of the physical switches are provided by zManager. Some TOR ports can be managed by the user from the zManager UI. The ESMs are not accessible for configuration changes through zManager's external management interfaces.

Metrics are provides for the following TOR port types:
- External - These ports that connect to customer's external network
- IEDN Host – These ports connect to System z IEDN network adapters, such as OSX.
- ISAOPT - These ports connect to ISAOPT Coordinator.
- BladeCenter ESM Ports – These ports connect to the ESM switches (other than ISAOPT).
- zBX to zBX- These ports connect the TOR in one zBX to the TOR in another zBX.

Metrics are provides for the following ESM port types:
- Uplinks to TOR – These ports connect the ESM to the TOR.
- Blade Ports – These ports connect the ESMs to the Blade network adapters.

Monitoring the physical switches can allow for determining the health and performance of the switches. For example, metrics such as dropped and discarded packets can affect the over all performance of workloads flowing through these switches. Bytes transferred metrics for ports provide the ability to determine bandwidth utilization.

Top-of-rack switch ports metrics

This metric collection group provides metrics for the Top-of-Rack (TOR) switch ports for the TORs in each zBX.

Metric Group Name
"top-of-rack-switch-ports"

Collection Interval
120 seconds

Applicable Managed Object Class
"zbx"

This metric collection provides metrics for an optimizer's physical network adapters.

Table 169. Top-of-rack switch port metrics group

Pos	Metric field name	Type	Units	Description
1	**switch-location-info**	String		The location of the switch in the zBX that was set by zManager. The switch-location-info is a 4 character cage location identifier. The first character identifies the zBX rack, and the remaining characters identify the vertical location within the rack
2	**port-num**	Integer		Switch port number
3	**type**	String Enum		• “B” (BladeCenter ESM port): attaches to an ESM switch. • “H” (Host port): attaches to System z network adapters for the IEDN • “E” (External port): attaches to external networking equipment • “I” (ISAOPT port): for ISAOPT • “Z” (zBX port): attaches the zBX to another zBX • “T” (TOR port): attaches to the other TOR switch in the same zBX
4	**remote-partner-info**	String		The remote partner depends upon the port type: • If type is “Z”: attached to a TOR in another zBX, this will be the location of the top-of-rack-switch. • If type is “E”: "N/A" • If type is “H”: This is the *cpc id.pchid* of the attached OSX. • If type is “B”: The format of the remote-partner-info is *bladecenter chassis-id_ esm location-id* The first 4 characters identify the blade center chassis within this zBX. The last 4 characters identify the ESM location ID • If type is “I”: The name of the blade center chassis containing the ISAOPT blades • If type is “T”: The name of the other TOR switch.
5	**bytes-sent**	Long	Count	Number of bytes sent from this port.
6	**bytes-received**	Long	Count	Number of bytes received to this port.
7	**packets-sent**	Long	Count	Number of packets sent from this port.
8	**packets-received**	Long	Count	Number of packets received to this port.
9	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending from this port.
10	**packets-received-dropped**	Long	Count	Number of packets received to this port that were dropped.
11	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from this port. Packets may be discarded due to errors associated with the packet, such as malformed packets.
12	**packets-received-discarded**	Long	Count	Number of packets received by these ports that were discarded. Packets may be discarded by the virtual or physical due to errors associated with the packet, such as malformed packets

Table 169. Top-of-rack switch port metrics group (continued)

Pos	Metric field name	Type	Units	Description
13	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from this port.
14	**multicast-packets-received**	Long	Count	Number of multicast packets received to this port.
15	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from this port.
16	**broadcast-packets-received**	Long	Count	Number of broadcast packets received to this port.
17	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this switch port over the collection interval.
18	**interval-bytes-received**	Long	Bytes	Number of bytes received by this switch port over the collection interval.
19	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this switch port over the collection interval.
20	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received per second by this switch port over the collection interval.
21	**flags**	Long		Flags indicating the types of metrics that are reported by this uplink. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

ESM switch port metrics

This metric group provides metrics for the ESM switch ports for each ESM in each zBX.

Metric Group Name
"ethernet-switch-module-ports"

Collection Interval
120 seconds

Applicable Managed Object Class
"zbx"

This metric collection provides metrics for an optimizer's physical network adapters.

Table 170. Optimizer IEDN physical network adapter metric group

Pos	Metric field name	Type	Units	Description
1	**switch-location-info**	String		The location of the switch in the zBX that was set by zManager. The switch-location-info is a 4 character cage location identifier. The first character identifies the zBX rack, and the remaining characters identify the vertical location within the rack.
2	**port-num**	Integer		Switch port number

Table 170. Optimizer IEDN physical network adapter metric group (continued)

Pos	Metric field name	Type	Units	Description
3	**type**	String Enum		• "I" (Internal port): This port is connected to a Blade. • "E" (External port): This port is connected to a TOR.
4	**remote-partner-info**	String		Information about the remote element connected to the port. The value depends on the **type** of the port. In the case where the port type is: • If type is "E": This field is the name of the top-of-rack-switch connected to this ESM port. This TOR resides in the same zBX as the ESM. • If type is "I": This field is the name of the attached blade. The blade resides in the same zBX and blade center chassis as the ESM.
5	**bytes-sent**	Long	Count	Number of bytes sent from this port.
6	**bytes-received**	Long	Count	Number of bytes received to this port.
7	**packets-sent**	Long	Count	Number of packets sent from this port.
8	**packets-received**	Long	Count	Number of packets received to this port.
9	**packets-sent-dropped**	Long	Count	Number of packets that were dropped when sending from this port.
10	**packets-received-dropped**	Long	Count	Number of packets received to this port that were dropped.
11	**packets-sent-discarded**	Long	Count	Number of packets that were discarded when sending from this port. Packets may be discarded due to errors associated with the packet, such as malformed packets.
12	**packets-received-discarded**	Long	Count	Number of packets received by these ports that were discarded. Packets may be discarded by the virtual or physical due to errors associated with the packet, such as malformed packets.
13	**multicast-packets-sent**	Long	Count	Number of multicast packets sent from this port.
14	**multicast-packets-received**	Long	Count	Number of multicast packets received to this port.
15	**broadcast-packets-sent**	Long	Count	Number of broadcast packets sent from this port.
16	**broadcast-packets-received**	Long	Count	Number of broadcast packets received to this port.
17	**interval-bytes-sent**	Long	Bytes	Number of bytes sent by this switch port over the collection interval.
18	**interval-bytes-received**	Long	Bytes	Number of bytes received by this switch port over the collection interval.
19	**bytes-per-second-sent**	Long	Bytes per Second	Number of bytes sent per second by this switch port over the collection interval.
20	**bytes-per-second-received**	Long	Bytes per Second	Number of bytes received per second by this switch port over the collection interval.

Table 170. Optimizer IEDN physical network adapter metric group (continued)

Pos	Metric field name	Type	Units	Description
21	**flags**	Long		Flags indicating the types of metrics that are reported by this uplink. The value of this field should be interpreted as a bitmask. The meaning of each bit is as follows: • 0x02 - Byte counts are supported • 0x04 – Packet counts are supported • 0x08 – Drop counts are supported • 0x10 – Discard counts are supported • 0x20 – Multicast counts are supported • 0x40 – Broadcast counts are supported • 0x80 – Interval bytes sent and received are supported

Appendix A. XML document structure of a performance policy

To import a performance policy for a workload resource group through the HMC **Workload Resource Group Details** task or the **Import Performance Policy** operation, you must first create an XML document that defines the policy elements. Use this topic to create a properly structured XML document to import.

The XML document starts with the **WorkloadPerformancePolicy** element, which contains all the elements of a workload resource group performance policy. The following sample shows the correct structure of the major elements in the **WorkloadPerformancePolicy** element:

```
<WorkloadPerformancePolicy
xmlns="http://www.ibm.com/PPM/WorkloadPerformancePolicy">
 <Name> SampleWorkloadPerformancePolicy </Name>
 <Description> Sample performance policy for a workload </Description>
 <Version> 3.00.00 </Version>
 <UI> PPM Editor </UI>
 <WorkloadImportance> High </WorkloadImportance>
 <ServiceClasses> ... </ServiceClasses>
</WorkloadPerformancePolicy>
```

The following table describes the major elements in the **WorkloadPerformancePolicy** element.

Table 171. Performance policy XML elements

Element name	Rqd/Opt	Description
Name	Required	The display name specified for the performance policy. All **Name** elements must be up to 64 characters long, consisting of alphanumeric characters, blanks, periods, underscores, or dashes. Names must start with an alphabetic character and end with an alphabetic character, numeric character, underscore, period, or dash. Names must be unique to other existing performance policies in the workload resource group.
Description	Optional	Arbitrary text describing the performance policy in up to 256 characters.
Version	Required	A version number, a release number, and a level number, which are each separated by a period. The version number must be between 1 and 99. The release and level numbers must be between 0 and 99.
UI	Required	The name of the product that last edited the XML document. **UI** has the same length and content restrictions as the **Name** element.
WorkloadImportance	Required	The importance value assigned to the performance policy, which is one of the following: **highest**, **high**, **medium**, **low**, or **lowest**
ServiceClasses	Required	The **ServiceClasses** element contains one or more service classes, as defined in "XML structure of a **ServiceClasses** element."

XML structure of a ServiceClasses element

The **ServiceClasses** element contains one or more **ServiceClass** elements that set the priority of and classify resources within a performance policy. A service class is a group of virtual servers that has the same service goals or performance objectives, resource requirements, or availability requirements.

The following sample shows the structure of a **ServiceClass** element within the **ServiceClasses** element:

```
<ServiceClasses>
 <ServiceClass>
  <Name> Web Hot </Name>
  <Description> Critical web transactions </Description>
  <Type> Server </Type>
```

```
  <Goal> ... </Goal>

  <RuleBuilderElement> ... </RuleBuilderElement>

 </ServiceClasses>
</ServiceClasses>
```

ServiceClass elements are positional. After you import and activate the policy, zManager searches service classes from low-ordered to high-ordered service class during classification to find a match.

The following table describes the elements in the **ServiceClass** element.

Table 172. Performance policy XML: Elements in a **ServiceClass** *element*

Element name	Rqd/Opt	Description
Name	Required	The name specified for the service class. The **Name** element must be a valid identifier up to 64 characters long, consisting of alphanumeric characters, blanks, periods, underscores, or dashes. Names must start with an alphabetic character and end with an alphabetic character, numeric character, underscore, period, or dash. Names must be unique to other existing service classes in the performance policy.
Description	Optional	Arbitrary text describing the service class in up to 256 characters.
Type	Required	This element identifies the resource associated with the service class. The **Type** value must be `server` to target specific virtual servers.
Goal	Required	The **Goal** element identifies the type of performance goal required for this service class, as described in "XML structure of a Goal element."
RuleBuilderElement	Required	The **RuleBuilderElement** contains the classification attributes for the service class, which are described in "XML structure of a RuleBuilderElement " on page 679.

XML structure of a Goal element

The **Goal** element identifies the type of performance goal, which is either a velocity goal or a discretionary goal. If you code the **Velocity** element, you also need to code the **Importance** element. The following samples show how to code the XML for each type of goal:

```
<Goal>
 <Discretionary/>
</Goal>
```

or

```
<Goal>
 <Velocity>
  <Importance> Medium </Importance>
  <Level> Fast </Level>
 </Velocity>
</Goal>
```

For every **Goal** element, you must code either the **Discretionary** element or the **Velocity** element. If you code the **Velocity** element, you must code the **Importance** and **Level** elements as described in the following table.

Table 173. Performance policy XML: Elements required for a **Velocity** *element*

Element name	Rqd/Opt	Description
Importance	Required	This field identifies the business importance level assigned to the service class, which must be one of the following: **highest**, **high**, **medium**, **low**, or **lowest**

Table 173. Performance policy XML: Elements required for a **Velocity** *element (continued)*

Element name	Rqd/Opt	Description
Level	Required	This field identifies the velocity goal value of the service class, which must be one of the following: **fastest**, **fast**, **moderate**, **slow**, or **slowest**

XML structure of a RuleBuilderElement

The **RuleBuilderElement** represents a classification rule for the service class. Each service class within a performance policy has one or more classification rules that identify how hardware or software elements of a workload resource group are associated with the service class. So each **ServiceClass** element can have one or more **RuleBuilderElement**s.

Each classification rule consists of one or more conditions that enable zManager to associate a service class to incoming work. A condition is a group containing a filter type, filter operator and filter value. To fully represent a classification rule, a **RuleBuilderElement** consists of one or more **Filter** elements, each containing one **FilterType**, one **FilterOperation**, and one **FilterValue** element.

Classification rules within a service class are positional. Assign any **RuleBuilderElement** containing a wildcard filter value to a high-order service class, and assign any **RuleBuilderElement** containing a specific filter value to a low-order service class.

The following sample shows the XML structure of a simple **RuleBuilderElement**:

```
<RuleBuilderElement>
  <RuleBuilderElementType> Rule </RuleBuilderElementType>
  <Filter>
    <FilterType> Virtual Server Name </FilterType>
    <FilterOperation> stringMatch </FilterOperation>
    <FilterValue> WebSales* </FilterValue
  </Filter>
</RuleBuilderElement>
```

This **RuleBuilderElement** classifies all virtual servers with the string "WebSales" in the name as resources to be managed according to the goals set for the service class.

All **RuleBuilderElement**s must begin with **RuleBuilderElementType**, which identifies the type of classification rule as one of the following:

Rule
: Defines a simple filter that resolves to true or false based on its filter pattern compared to a specified virtual server attribute.

And
: Defines a complex set of two rules; during classification, both of the two rules imbedded within this **RuleBuilderElement** must be true for the virtual server to be managed according to the goals for this service class

Or Defines a complex set of two rules; during classification, only one of the two rules imbedded within this **RuleBuilderElement** must be true for the virtual server to be managed according to the goals for this service class

If you code a **RuleBuilderElementType** value of **and** or **or**, exactly two **RuleBuilderElement**s must be nested inside this **RuleBuilderElement** object so they can be logically compared. The following sample shows nested **RuleBuilderElement**s for a complex set of rules that are equivalent to the expression `Rule 1 and (Rule 2 or Rule 3)`:

```
<RuleBuilderElement>

  <RuleBuilderElementType>And</RuleBuilderElementType>
```

```
  <RuleBuilderElement>
    <RuleBuilderElementType>Rule</RuleBuilderElementType>
    <Filter> ... filters for rule 1 ... </Filter>
  </RuleBuilderElement>

  <RuleBuilderElement>
    <RuleBuilderElementType>Or</RuleBuilderElementType>

    <RuleBuilderElement>
       <RuleBuilderElementType>Rule</RuleBuilderElementType>
       <Filter> ... filters for rule 2 ... </Filter>
    </RuleBuilderElement>

    <RuleBuilderElement>
       <RuleBuilderElementType>Rule</RuleBuilderElementType>
       <Filter> ... filters for rule 3 ... </Filter>
    </RuleBuilderElement>

  </RuleBuilderElement>

</RuleBuilderElement>
```

Within every **Filter** element, you must code one **FilterType**, one **FilterOperation**, and one **FilterValue** element. These elements are described in the following table.

Table 174. Performance policy XML: Elements in a **Filter** *element*

Element name	Rqd/Opt	Description
FilterType	Required	This element identifies the virtual server attribute to be used during classification. Valid values are: **`Hostname`** The host name of the virtual server. To use this value, a guest platform management provider must be running on the operating system on the virtual server. **`Virtual Server Name`** The virtual server ID for z/VM and x Hyp or the logical partition (LPAR) ID for PowerVM and PR/SM. This ID is the same as the name used on the **Virtual Servers** tab on the **Ensemble Management** window in the primary HMC. **`OS Type`** The type of operating system, such as AIX® or Linux, that is running on the virtual server. You can select this type only if he virtual server definition contains the OS type. To use this value, a guest platform management provider must be running on the operating system on the virtual server. **`OS Level`** The release level of the operating system running on the virtual server. To use this value, a guest platform management provider must be running on the operating system on the virtual server. **`OS Name`** The name of the operating system image running on the virtual server as known to its operating system. To use this value, a guest platform management provider must be running on the operating system on the virtual server.

Table 174. Performance policy XML: Elements in a **Filter** *element (continued)*

Element name	Rqd/Opt	Description
FilterOperation	Required	This element identifies the logical filter operation, which must be one of the following: • **stringMatch** – the filter value must match the property defined by the filter type • **stringNotMatch** – the filter value must not match the property defined by the filter type
FilterValue	Required	The value to be used for classification. Possible values vary based on the value specified for the **FilterType** element. A filter value cannot be longer than 255 characters. A filter value must be a regular expression that can include one period (.) as a substitute for one character and one wildcard (.*) as a substitute for multiple characters. This wildcard can be used only at the end of an expression. • For filter type **Hostname**, provide a fully qualified host name that consists of the host name and the TCP domain name. The TCP domain name specifies a group of systems that share a common suffix (domain name). For example, given a fully qualified host name of `server1.us.ibm.com`: – `server1` is the host name – `us.ibm.com` is the TCP domain name The fully qualified host name `server1.us.ibm.com` illustrates a character-based format, but you can use a numerical-based name. The numerical-based host name is not the same as the system's IP address. • For filter type **OS Level**, provide the release level of the operating system. • For filter type **OS Name**, provide the LPAR name or virtual machine ID. • For filter type **OS Type**, provide the name of the operating system. • For filter type **Virtual Server Name**, provide the virtual server name shown in the **Virtual Server Details** window in the HMC.

Sample XML document for a performance policy

The following sample illustrates a correctly structured XML document for a workload resource group performance policy.

This performance policy consists of two service classes:

SC1
This service class sets a performance goal of slow velocity and medium importance for virtual servers with attributes that match both of the following classification rules:
- The virtual server name is “Server1”, and
- The hostname is “abc.pok.ibm.com”

SC2
This service class sets a performance goal of slow velocity and medium importance for virtual servers with a virtual server name that matches “Lpar1”.

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkloadPerformancePolicy
 xmlns="http://www.ibm.com/PPM/WorkloadPerformancePolicy">

 <Name>Workload Policy name</Name>
 <Description>Workload performance policy for sample workload</Description>
 <Version>1.00.00</Version>
 <UI>PPM Editor</UI>
 <WorkloadImportance>High</WorkloadImportance>

 <ServiceClasses>

  <ServiceClass>
   <Name>SC1</Name>
   <Description>SC1 Description</Description>
   <Type>Server</Type>

   <RuleBuilderElement>
    <RuleBuilderElementType>And</RuleBuilderElementType>
     <RuleBuilderElement>
       <RuleBuilderElementType>Rule</RuleBuilderElementType>
       <Filter>
        <FilterType>Virtual Server Name</FilterType>
        <FilterOperation>stringMatch</FilterOperation>
        <FilterValue>Server1</FilterValue>
       </Filter>
     </RuleBuilderElement>
     <RuleBuilderElement>
      <RuleBuilderElementType>Rule</RuleBuilderElementType>
       <Filter>
        <FilterType>Hostname</FilterType>
        <FilterOperation>stringMatch</FilterOperation>
        <FilterValue>abc.pok.ibm.com</FilterValue>
       </Filter>
     </RuleBuilderElement>
   </RuleBuilderElement>

   <Goal>
    <Importance>Medium</Importance>
    <Velocity>Slow</Velocity>
   </Goal>
  </ServiceClass>
```

Figure 301. Policy XML example, Part 1

```
  <ServiceClass>
   <Name>SC2</Name>
   <Description>Service Class for virtual server</Description>
   <Type>Server</Type>

   <RuleBuilderElement>
    <RuleBuilderElementType>Rule</RuleBuilderElementType>
    <Filter>
     <FilterType>Virtual Server Name</FilterType>
     <FilterOperation>stringMatch</FilterOperation>
     <FilterValue>Lpar1</FilterValue>
    </Filter>
   </RuleBuilderElement>

   <Goal>
    <Importance>Medium</Importance>
    <Velocity>Slow</Velocity>
   </Goal>
  </ServiceClass>

 </ServiceClasses>
</WorkloadPerformancePolicy>
```

Figure 302. Policy XML example, Part 2

Appendix B. Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 USA

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION " AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml.

Intel, and Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linux Torvalds in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Electronic emission notices

The following statements apply to this IBM product. The statement for other IBM products intended for use with this product will appear in their accompanying manuals.

Federal Communications Commission (FCC) Statement

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions contained in the installation manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Properly shielded and grounded cables and connectors must be used in order to meet FCC emission limits. IBM is not responsible for any radio or television interference caused by using other than recommended cables and connectors, by installation or use of this equipment other than as specified in the installation manual, or by any other unauthorized changes or modifications to this equipment. Unauthorized changes or modifications could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Canadian Department of Communications Compliance Statement

This Class A digital apparatus complies with Canadian ICES-003.

Avis de conformlté aux normes du ministère des Communications du Canada

Cet appareil numérique de la classe A est conform à la norme NMB-003 du Canada.

European Union (EU) Electromagnetic Compatibility Directive

This product is in conformity with the protection requirements of EU Council Directive 2004/108/EC on the approximation of the laws of the Member States relating to electromagnetic compatibility. IBM cannot accept responsibility for any failure to satisfy the protection requirements resulting from a non-recommended modification of the product, including the fitting of non-IBM option cards.

This product has been tested and found to comply with the limits for Class A Information Technology Equipment according to European Standard EN 55022. The limits for Class equipment were derived for commercial and industrial environments to provide reasonable protection against interference with licensed communication equipment.

Warning: This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may be required to take adequate measures.

European Community contact:
IBM Deutschland GmbH
Technical Regulations, Department M372
IBM-Allee 1, 71139 Ehningen, Germany
Telephone: 0049 (0) 7032 15-2941
email: lugi@de.ibm.com

EC Declaration of Conformity (In German)

Deutschsprachiger EU Hinweis: Hinweis für Geräte der Klasse A EU-Richtlinie zur Elektromagnetischen Verträglichkeit

Dieses Produkt entspricht den Schutzanforderungen der EU-Richtlinie 89/336/EWG zur Angleichung der Rechtsvorschriften über die elektromagnetische Verträglichkeit in den EU-Mitgliedsstaaten und hält die Grenzwerte der EN 55022 Klasse A ein.

Um dieses sicherzustellen, sind die Geräte wie in den Handbüchern beschrieben zu installieren und zu betreiben. Des Weiteren dürfen auch nur von der IBM empfohlene Kabel angeschlossen werden. IBM übernimmt keine Verantwortung für die Einhaltung der Schutzanforderungen, wenn das Produkt ohne Zustimmung der IBM verändert bzw. wenn Erweiterungskomponenten von Fremdherstellern ohne Empfehlung der IBM gesteckt/eingebaut werden.

EN 55022 Klasse A Geräte müssen mit folgendem Warnhinweis versehen werden:

"Warnung: Dieses ist eine Einrichtung der Klasse A. Diese Einrichtung kann im Wohnbereich Funk-Störungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen zu ergreifen und dafür aufzukommen."

Deutschland: Einhaltung des Gesetzes über die elektromagnetische Verträglichkeit von Geräten

Dieses Produkt entspricht dem “Gesetz über die elektromagnetische Verträglichkeit von Geräten (EMVG)“. Dies ist die Umsetzung der EU-Richtlinie 89/336/EWG in der Bundesrepublik Deutschland.

Zulassungsbescheinigung laut dem Deutschen Gesetz über die elektromagnetische Verträglichkeit von Geräten (EMVG) vom 18. September 1998 (bzw. der EMC EG Richtlinie 89/336) für Geräte der Klasse A.

Dieses Gerät ist berechtigt, in Übereinstimmung mit dem Deutschen EMVG das EG-Konformitätszeichen - CE - zu führen.

Verantwortlich für die Konformitätserklärung nach Paragraf 5 des EMVG ist die IBM Deutschland GmbH, 70548 Stuttgart.

Informationen in Hinsicht EMVG Paragraf 4 Abs. (1) 4:

Das Gerät erfüllt die Schutzanforderungen nach EN 55024 und EN 55022 Klasse A.

update: 2004/12/07

People's Republic of China Class A Compliance Statement

This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may need to perform practical actions.

声　明

此为A级产品，在生活环境中，
该产品可能会造成无线电干扰。
在这种情况下，可能需要用户对其
干扰采取切实可行的措施。

Japan Class A Compliance Statement

This is a Class A product based on the standard of the VCCI Council. If this equipment is used in a domestic environment, radio interference may occur, in which case, the user may be required to take corrective actions.

この装置は, クラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。 VCCI－A

Korean Class A Compliance Statement

이 기기는 업무용(A급)으로 전자파적합등록을 한 기기이오니
판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의
지역에서 사용하는 것을 목적으로 합니다.

Taiwan Class A Compliance Statement

Warning: This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user will be required to take adequate measures.

警告使用者:
這是甲類的資訊產品，在
居住的環境中使用時，可
能會造成射頻干擾，在這
種情況下，使用者會被要
求採取某些適當的對策。

台灣IBM 產品服務聯絡方式：
台灣國際商業機器股份有限公司
台北市松仁路7號3樓
電話：0800-016-888

Glossary

A.

advanced management module (AMM). A hardware unit that provides system-management functions for all the blade servers in a BladeCenter chassis.

alternate HMC. A System z Hardware Management Console (HMC) that is paired with the primary HMC to provide redundancy.

See also primary HMC.

AMM. See advanced management module.

appliance. A software device that provides a narrow range of functions and generally runs on a hardware platform.

application environment. The environment that includes the software and the server or network infrastructure that supports it.

ARM-instrumented application. An application in which application response measurement (ARM) calls are added to the source code so that management systems can monitor the performance of the application. ARM is an Open Group standard.

Automate suite (Automate). The second of two suites of functionality associated with the IBM zEnterprise Unified Resource Manager. The Automate suite includes goal-oriented monitoring and management of resources and energy management.

See also Manage suite.

B.

blade. A hardware unit that provides application-specific services and components. The consistent size and shape (or form factor) of each blade allows it to fit in a BladeCenter chassis.

BladeCenter chassis. A modular chassis that can contain multiple blades, allowing the individual blades to share resources such as the management, switch, power, and blower modules.

C.

central processor complex (CPC). A physical collection of hardware that consists of main storage, one or more central processors, timers, and channels. In the zEnterprise environment, the CPC consists of a zEnterprise mainframe and any attached IBM zEnterprise BladeCenter Extension (zBX).

See also node and zCPC.

classification rule. A rule used by System z workload resource group manager firmware and software to assign a service class.

CPC. See central processor complex.

D.

DataPower XI50z. See IBM WebSphere® DataPower Integration Appliance XI50 for zEnterprise.

discretionary goal. A service class performance goal assigned to low priority work that does not have any specific performance goal. Work is run when system resources are available.

E.

ensemble. A collection of one or more zEnterprise nodes (including any attached zBX) that are managed as a single logical virtualized system by the Unified Resource Manager, through the Hardware Management Console.

ensemble member. A zEnterprise node that has been added to an ensemble.

See also node.

F.

firmware. Licensed Internal Code (LIC) that is shipped with hardware. Firmware is considered an integral part of the system and is loaded and run at power on. Firmware is not open for customer configuration and is expected to run without any customer setup.

G.

GPMP. See guest platform management provider.

guest platform management provider (GPMP). An optional suite of applications that is installed in specific z/OS, Linux, and AIX operating system images to support platform management functions. For example, the guest platform management provider collects and aggregates performance data for virtual servers and workload resource groups.

H.

Hardware Management Console (HMC). A user interface through which data center personnel configure, control, monitor, and manage System z hardware and software resources. The HMC communicates with each central processor complex (CPC) through the Support Element. On an IBM zEnterprise 196 (z196), using the Unified Resource

Manager on the HMCs or Support Elements, personnel can also create and manage an ensemble.

See also primary HMC and alternate HMC.

HMC. See Hardware Management Console.

hypervisor. A program that allows multiple instances of operating systems or virtual servers to run simultaneously on the same hardware device. A hypervisor can run directly on the hardware, can run within an operating system, or can be imbedded in platform firmware. Examples of hypervisors include PR/SM, z/VM, and PowerVM Enterprise Edition.

I.

IBM blade. A customer-acquired, customer-installed select blade to be managed by IBM zEnterprise Unified Resource Manager. One example of an IBM blade is a POWER7 blade.

IBM System z Advanced Workload Analysis Reporter (IBM zAware). Firmware consisting of an integrated set of applications that monitor software running on z/OS and model normal system behavior. IBM zAware pattern recognition techniques identify unexpected messages, providing rapid diagnosis of problems caused by system changes. Operational controls and views of analytical data are available through the IBM zAware graphical user interface (GUI).

IBM System z Application Assist Processor (zAAP). A specialized processor that provides a Java execution environment, which enables Java-based web applications to be integrated with core z/OS business applications and backend database systems.

IBM System z Integrated Information Processor (zIIP). A specialized processor that provides computing capacity for selected data and transaction processing workloads and for selected network encryption workloads.

IBM WebSphere DataPower Integration Appliance XI50 for zEnterprise (DataPower XI50z). A purpose-built appliance that simplifies, helps secure, and optimizes XML and Web services processing.

IBM zAware. See IBM System z Advanced Workload Analysis Reporter (IBM zAware).

IBM zAware disaster recovery environment. An IBM zAware environment that is created expressly for disaster recovery purposes.

IBM zAware environment. A configuration that consists of an IBM zAware partition and the IBM zAware monitored clients that are sending information to the IBM zAware server that is running on the partition. The IBM zAware monitored clients do not have to run in the same IBM zAware host system that contains the partition.

IBM zAware host system. The zEC12 central processor complex (CPC) that contains the logical partition (LPAR) in which the IBM System z Advanced Workload Analysis Reporter (IBM zAware) runs.

IBM zAware model. A description of normal behavior that an IBM zAware server generates for a specific monitored z/OS system. Initially, this model is based on prior data (system and application messages) from the operations log (OPERLOG) for the z/OS system. The model is updated periodically and can be modified to include or exclude specific days of system operation. The IBM zAware server uses this model to detect system problems that are indicated in current data that the server receives from the specific z/OS system.

IBM zAware monitored client. A z/OS partition that sends OPERLOG logstream data to the IBM System z Advanced Workload Analysis Reporter (IBM zAware) for analysis. To detect problems, IBM zAware compares the system and application messages in these log files to a model of normal behavior for this z/OS system, and highlights anomalous results through the IBM zAware graphical user interface (GUI).

IBM zAware partition. The logical partition (LPAR) in the zEC12 central processor complex (CPC) in which only an IBM zAware server runs. The IBM zAware graphical user interface (GUI) provides operational controls and views of analytical data for IBM zAware monitored clients.

IBM zAware server. An instance of the IBM System z Advanced Workload Analysis Reporter (IBM zAware) that is receiving data from monitored clients.

IBM zEnterprise 114 (z114). The newest generation of the entry System z family of servers built on a new processor chip, featuring a 14-way core design with enhanced memory function and capacity, security, and on demand enhancements to support existing mainframe workloads and consolidation.

IBM zEnterprise 196 (z196). The previous generation of the System z high end family of servers built on a new processor chip, featuring a 96-way core design with enhanced memory function and capacity, security, and on demand enhancements to support existing mainframe workloads and large scale consolidation.

IBM zEnterprise BladeCenter Extension (zBX). A heterogeneous hardware infrastructure that consists of a BladeCenter chassis attached to a zEC12, z196, or z114. A BladeCenter chassis can contain IBM blades or optimizers.

IBM zEnterprise BladeCenter Extension (zBX) blade. Generic name for all blade types supported in an IBM zEnterprise BladeCenter Extension (zBX). This term includes IBM blades and optimizers.

IBM zEnterprise EC12 (zEC12). The newest generation of the System z high end family of servers

built on a new processor chip, featuring a 120-way core design with enhanced memory function and capacity, security, and on demand enhancements to support existing mainframe workloads and large scale consolidation.

IBM zEnterprise System (zEnterprise). A heterogeneous hardware infrastructure that can consist of a zEC12, z196, or z114 and an attached IBM zEnterprise BladeCenter Extension (zBX), managed as a single logical virtualized system by the Unified Resource Manager.

IBM zEnterprise Unified Resource Manager. Licensed Internal Code (LIC), also known as firmware, that is part of the Hardware Management Console. The Unified Resource Manager provides energy monitoring and management, goal-oriented policy management, increased security, virtual networking, and data management for the physical and logical resources of a given ensemble.

IEDN. See intraensemble data network (IEDN).

IEDN TOR switch. See intraensemble data network (IEDN) TOR switch.

INMN. See intranode management network (INMN).

intraensemble data network (IEDN). A private high-speed network for application data communications within an ensemble. Data communications for workload resource groups can flow over the IEDN within and between nodes of an ensemble. The Unified Resource Manager configures, provisions, and manages all of the physical and logical resources of the IEDN.

intraensemble data network (IEDN) TOR switch. A top-of-rack switch that provides connectivity to the intraensemble data network (IEDN), supporting application data within an ensemble.

intranode management network (INMN). A private service network that the Unified Resource Manager uses to manage the resources within a single zEnterprise node. The INMN connects the Support Element to the zEC12, z196, or z114 and to any attached IBM zEnterprise BladeCenter Extension (zBX).

M.

Manage suite (Manage). The first suite of functionality associated with the IBM zEnterprise Unified Resource Manager. The Manage suite includes core operational controls, installation, and configuration management, and energy monitoring.

management TOR switch. A top-of-rack switch that provides a private network connection between a zEC12, z196, or z114 Support Element and an IBM zEnterprise BladeCenter Extension (zBX).

member. See ensemble member.

N.

network interface card (NIC). A printed circuit board that plugs into a server. It controls the exchange of data over a network and provides the electronic functions for the data link protocol or access method, such as token ring or Ethernet.

NIC. See network interface card.

node. A single zEC12, z196, or z114 and any optionally attached IBM zEnterprise BladeCenter Extension (zBX). A node can be a member of only one ensemble.

See also central processor complex.

O.

optimizer. A special-purpose hardware component or appliance that can perform a limited set of specific functions with optimized performance when compared to a general-purpose processor. Because of its limited set of functions, an optimizer is an integrated part of a processing environment, rather than a standalone unit.

One example of an optimizer is the IBM WebSphere DataPower Integration Appliance XI50 for zEnterprise.

out-of-band monitoring solution. A type of monitoring solution that runs on a dedicated server rather than relying on the use of a monitoring agent installed in the operating system. For example, the IBM System z Advanced Workload Analysis Reporter (IBM zAware) provides out-of-band monitoring because it runs in a dedicated PR/SM partition and monitors clients that run in other partitions in System z servers.

OSM. An OSA-Express channel path identifier (CHPID) type that provides connectivity to the intranode management network (INMN).

OSX. An OSA-Express channel path identifier (CHPID) type that provides connectivity to the intraensemble data network (IEDN).

P.

performance index. A number that indicates whether the performance goal for a service class was achieved, exceeded, or missed.

performance policy. A description of the performance objectives and importance of a workload resource group.

platform management. The subset of systems management focused on hardware and virtualization management.

PowerVM. See PowerVM Enterprise Edition.

PowerVM Enterprise Edition (PowerVM). A hypervisor that provides a set of comprehensive systems technologies and services designed to enable aggregation and management of IBM POWER blade resources through a consolidated, logical view.

primary HMC. The System z Hardware Management Console (HMC) through which data personnel create and manage an ensemble. This HMC owns configuration and policy information that the Unified Resource Manager uses to monitor, manage, and adjust resources for all members of this ensemble.

See also alternate HMC.

private system control network (PSCN). The private subsystem of the System z servers that is controlled by a fully redundant dual-Ethernet communications network. This network provides communication to all field-replaceable units (FRUs) and hierarchic control through a mirrored system of control cards and IP addresses. The PSCN provides a means for subsystems to communicate and control the dynamic parameters of system operation. The PSCN also supports error reporting, failure data collection and recovery detection, and correction of both the internal hardware and firmware of the System z servers.

PSCN. See private system control network.

R.

rack. A free-standing structure or frame that can hold multiple servers and expansion units, such as BladeCenter blades.

response time goal. A service class performance goal that defines end-to-end response time of work requests.

S.

service class. A collection of work that has the same service goals or performance objectives, resource requirements, or availability requirements.

static power save mode. A zEC12, z196, or z114 function used for periods of low utilization or potentially when a CBU system is sitting idle waiting to take over in the event of a failure. The server uses frequency and voltage reduction to reduce energy consumption of the system. The customer initiates static power save mode by using the HMC or Support Element or Active Energy Manager.

T.

top-of-rack (TOR) switch. A network switch that is located in the first rack of an IBM zEnterprise BladeCenter Extension (zBX).

TOR switch. See intraensemble data network (IEDN) TOR switch and management TOR switch.

transaction. A unit of processing consisting of one or more application programs, affecting one or more objects, that is initiated by a single request.

U.

Unified Resource Manager. See IBM zEnterprise Unified Resource Manager.

V.

velocity goal. A service class performance goal that defines the acceptable amount of delay for work when work is ready to run. Velocity is the measure of how fast work should run when ready, without being delayed by contention for managed resources.

virtual appliance. A prepackaged software application that provides some well-defined business workflow, making it easier to deploy a solution with minimal configuration. Many tiers of operating system and applications can be packaged as a single virtual appliance. These tiers can depend on the hardware resources of different architectures.

See also virtual server collection and virtual server image.

virtual server. A logical construct that appears to comprise processor, memory, and I/O resources conforming to a particular architecture. A virtual server can support an operating system, associated middleware, and applications. A hypervisor creates and manages virtual servers.

virtual server collection. A set of virtual servers that supports a workload resource group. This set is not necessarily static. The constituents of the collection at any given point are determined by the virtual servers involved in supporting the workload resource group at that time.

See also virtual appliance and virtual server image.

virtual server image. A package containing metadata that describes the system requirements, virtual storage drives, and any goals and constraints for the virtual machine (for example, isolation and availability). The Open Virtual Machine Format (OVF) is a Distributed Management Task Force (DMTF) standard that describes a packaging format for virtual server images.

See also virtual appliance and virtual server collection.

virtual server image capture. The ability to store metadata and disk images of an existing virtual server. The metadata describes the virtual server storage, network needs, goals, and constraints. The captured information is stored as a virtual server image that can be referenced and used to create and deploy other similar images.

virtual server image clone. The ability to create an identical copy (clone) of a virtual server image that can be used to create a new similar virtual server.

W.

workload. The amount of application processing that a computer performs at a given time. In z/OS WLM, a workload is a customer-defined collection of work to be tracked, managed, and reported as a unit. For zEnterprise, see workload resource group.

workload resource group. A collection of virtual servers that perform a customer-defined collective purpose. A workload resource group generally can be viewed as a multi-tiered application. Each workload resource group is associated with a set of policies that define performance goals.

Z.

z/VM single system image (SSI) cluster. A collection of z/VM systems (called members) that can be managed, serviced, and administered as one system within which workloads can be deployed. An SSI cluster is intended to share a set of resources among all members.

z114. See IBM zEnterprise 114 (z114).

z196. See IBM zEnterprise 196 (z196).

zEC12. See IBM zEnterprise EC12 (zEC12).

zAAP. See IBM System z Application Assist Processor.

zBX. See IBM zEnterprise BladeCenter Extension (zBX).

zBX blade. See IBM zEnterprise BladeCenter Extension (zBX) blade.

zCPC. The physical collection of main storage, central processors, timers, and channels within a zEnterprise mainframe. Although this collection of hardware resources is part of the larger zEnterprise central processor complex, you can apply energy management policies to the zCPC that are different from those that you apply to any attached IBM zEnterprise BladeCenter Extension (zBX) or blades.

See also central processor complex.

zIIP. See IBM System z Integrated Information Processor.

zEnterprise. See IBM zEnterprise System (zEnterprise).

Unified Resource Manager. See IBM zEnterprise Unified Resource Manager.

Index

M

P

Q

R

S

U

W

X

Printed in USA

SC27-2617-01