Netcool/OMNIbus
Version 7 Release 4

Web GUI Administration API (WAAPI) User's Guide



SC14-7535-00

Netcool/OMNIbus
Version 7 Release 4

Web GUI Administration API (WAAPI) User's Guide



SC14-7535-00

Note
Before using this information and the product it supports, read the information in Notices.

This edition applies to version 7, release 4 of IBM Tivoli Netcool/OMNIbus (product number 5724-S44) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2011, 2012.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

What this publication contains v
Intended audience v
What this publication contains v
Publications vi
Accessibility viii
Tivoli technical training viii
Support information viii
Conventions used in this publication viii

Chapter 1. The Web GUI Administration API (WAAPI) 1
About WAAPI 1
Comparative procedures for modifying the Web GUI 2
Communications between the WAAPI client and the Web GUI server 3
Security 4

Chapter 2. Using WAAPI 5
Setting up the WAAPI properties file 5
Sending WAAPI requests to the server 5

Chapter 3. WAAPI requests 7
Structure of a WAAPI request 7
Characteristics of WAAPI XML documents . . . 9
Sample Requests 11
User requests 11
Maintain users 11
Modify a user 11
Get a list of users 19
View requests 19
Create a view 19
Create or replace a view 24
Modify a view 25
Delete a view 26
Get a list of views 26
Map requests 26
Create a map 27
Create or replace a map 43
Delete a map 43
Get a list of maps 44
Modify a map 44
Add a map visual 44
Add or replace a map visual 45
Delete a map visual 45
Modify a map visual 46
Resource requests 46
Add a resource 46
Create or replace a resource 47
Remove a resource 48
Get a list of resources 48
File requests 48
Add a directory 48
Add a file 49
Create or replace a file 50
Delete a file 50
Remove a directory 50
Recursively remove a directory 51
Menu requests 51
Create a menu 51
Create or replace a menu 53
Delete a menu 54
Get a list of menus 54
Modify a menu 54
Tool requests 55
Create a tool 55
Create or replace a tool 62
Delete a tool 62
Get a list of tools 63
Modify a tool 63
Prompt requests 64
Create or replace a prompt 64
Delete a prompt 65
Get a list of prompts 65
Modify a prompt 65
CGI requests 66
Register a CGI script 66
Create or replace a CGI script 67
Modify a CGI script 67
Unregister a CGI script 67
Filter requests 68
Add a filter 68
Create or replace a filter 72
Delete a filter 73
Get a list of filters 73
Modify a filter 73
Set the default view 74
Filter collection requests 74
Create a filter collection 75
Add a filter to a filter collection 76
Create or replace a filter collection 76
Delete a filter collection 77
Delete a filter from a filter collection 77
Get a list of filter collections 77
Modify a filter collection 78
Set the view for a filter collection 78
Metric requests 79
Create a metric 79
Create or replace a metric 83
Delete a metric 84
Get a list of metrics 84
Modify a metric 85
Relationship requests 86
Create a relationship 86
Create or replace a relationship 88
Delete a relationship 89
Get a list of relationships 89
Modify a relationship 89
Other requests 90
Resynchronize the Web GUI cache with the ObjectServer's database 90

© Copyright IBM Corp. 2011, 2012

Remove a node from a load balancing cluster . . 90
Generate a system status report. 91
Reload filters and views 91
Generate a system status report. 91
Reload filters and views 92
Remove a node from a load balancing cluster . . 92
Resynchronize the Web GUI cache with the ObjectServer's database 92

Appendix A. WAAPI properties and command-line options 93

Appendix B. Installing the WAAPI client on a remote host 97

Appendix C. WAAPI security 99
Creating secure WAAPI connections 99
Creating a WAAPI SSL connection (server-only authentication) 100
Creating a WAAPI SSL connection (client-server authentication) 101
Creating a WAAPI SSL connection with FIPS 140–2 (server-only authentication) 102
Creating a WAAPI SSL connection with FIPS 140–2 (client-server authentication) 104
Enabling WAAPI password encryption 106
Encrypting WAAPI passwords using AES . . . 106
Encrypting WAAPI passwords using FIPS 140–2 mode encryption 107
Securing the waapi.init properties file 108

Notices 109
Trademarks 112

Index 113

What this publication contains

The IBM Tivoli Netcool/OMNIbus Web GUI is a Web-based application that processes network events from one or more data sources and presents the event data to users in various graphical formats.

The *IBM Tivoli Netcool/OMNIbus Web GUI Administration API (WAAPI) User's Guide* shows how to administer the Tivoli Netcool/OMNIbus Web GUI using an XML application programming interface named WAAPI.

Intended audience

This publication is intended for administrators of the Tivoli Netcool/OMNIbus Web GUI. The publication provides information on how to administer the Web GUI using WAAPI.

This publication assumes that the administrator has a reasonable degreee of XML knowledge. In particular, the publication assumes that the administrator understands the following concepts:

- The rules, logic, and components that are used by XML
- The concepts of *elements*, *attributes*, and *markup*
- How to create documents that are well-formed, and valid against an XML document type definition (DTD)

What this publication contains

This publication contains the following sections:

- Chapter 1, "The Web GUI Administration API (WAAPI)," on page 1

  An introduction to the WAAPI facilities and how it interacts with the Web GUI server.
- Chapter 2, "Using WAAPI," on page 5

  How to use the WAAPI client to send administration requests to the Web GUI server.
- Chapter 3, "WAAPI requests," on page 7

  Defines the structure and content of each type of WAAPI request. The section also includes examples of each type of request.
- Appendix A, "WAAPI properties and command-line options," on page 93

  The properties and their equivalent command line options that determine the behavior of WAAPI.
- Appendix B, "Installing the WAAPI client on a remote host," on page 97

  Instructions on how to install the WAAPI client on a node remote from the Web GUI server.
- Appendix C, "WAAPI security," on page 99

  Instructions on how to use the security features of WAAPI to create secure connections to the Web GUI server, encrypt passwords, and secure the WAAPI properties file.

© Copyright IBM Corp. 2011, 2012

Publications

This section lists publications in the Tivoli Netcool/OMNIbus library and related documents. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

Your Tivoli Netcool/OMNIbus library

The following documents are available in the Tivoli Netcool/OMNIbus library:

- *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*, SC14-7526

  Includes installation and upgrade procedures for Tivoli Netcool/OMNIbus, and describes how to configure security and component communications. The publication also includes examples of Tivoli Netcool/OMNIbus architectures and describes how to implement them.
- *IBM Tivoli Netcool/OMNIbus Administration Guide*, SC14-7527

  Describes how to perform administrative tasks using the Tivoli Netcool/OMNIbus Administrator GUI, command-line tools, and process control. The publication also contains descriptions and examples of ObjectServer SQL syntax and automations.
- *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide*, SC14-7528

  Describes how to perform administrative and event visualization tasks using the Tivoli Netcool/OMNIbus Web GUI.
- *IBM Tivoli Netcool/OMNIbus User's Guide*, SC14-7529

  Provides an overview of the desktop tools and describes the operator tasks related to event management using these tools.
- *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*, SC14-7530

  Contains introductory and reference information about probes and gateways, including probe rules file syntax and gateway commands.
- *IBM Tivoli Monitoring for Tivoli Netcool/OMNIbus Agent User's Guide*, SC14-7532

  Describes how to install the health monitoring agent for Tivoli Netcool/OMNIbus and contains reference information about the agent.
- *IBM Tivoli Netcool/OMNIbus Event Integration Facility Reference*, SC14-7533

  Describes how to develop event adapters that are tailored to your network environment and the specific needs of your enterprise. This publication also describes how to filter events at the source.
- *IBM Tivoli Netcool/OMNIbus Error Messages Guide*, SC14-7534

  Describes system messages in Tivoli Netcool/OMNIbus and how to respond to those messages.
- *IBM Tivoli Netcool/OMNIbus Web GUI Administration API (WAAPI) User's Guide*, SC22-7535

  Shows how to administer the Tivoli Netcool/OMNIbus Web GUI using the XML application programming interface named WAAPI
- *IBM Tivoli Netcool/OMNIbus ObjectServer HTTP Interface Reference Guide*, SC27-5613Describes the URIs and common behaviors of the Application Programming Interface (API) that is called the ObjectServer HTTP Interface. Describes how to enable the API and provides examples of JSON payloads, and HTTP requests and responses.
- *IBM Tivoli Netcool/OMNIbus ObjectServer OSLC Interface Reference Guide*, SC27-5613Describes the services, resources, and common behaviors of the Open Services for Lifecycle Collaboration (OSLC) Application Programming Interface

(API) that is called the ObjectServer OSLC Interface. Describes how to enable the API and provides examples of service provider definitions, RDF/XML payloads, and HTTP requests and responses.

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at the following Web address:

http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Information Center Web site at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File** > **Print** window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at the following Web site:

http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to the following Web site:
   http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss
2. Select your country from the list and click **Go**. The Welcome to the IBM Publications Center page is displayed for your country.
3. On the left side of the page, click **About this site** to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully.

With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate most features of the graphical user interface.

For additional information, see the Accessibility Appendix in Accessibility features for the Web GUI.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site:

http://www.ibm.com/software/tivoli/education

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online
Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant
The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, go to http://www.ibm.com/software/support/isa

Documentation
If you have a suggestion for improving the content or organization of this guide, send it to the Tivoli Netcool/OMNIbus Information Development team at:

mailto://L3MMDOCS@uk.ibm.com

Conventions used in this publication

This publication uses several conventions for special terms and actions and operating system-dependent commands and paths.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes,

multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:** and **Operating system considerations:**)

- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs)
- Words defined in text (example: a nonswitched line is called a *point-to-point* line)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data
- Variables and values you must provide: ... where *myname* represents....

Monospace

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace **$***variable* with **%***variable***%** for environment variables, and replace each forward slash (/) with a backslash (\) in directory paths. For example, on UNIX systems, the $NCHOME environment variable specifies the path of the Netcool® home directory. On Windows systems, the %NCHOME% environment variable specifies the path of the Netcool home directory. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Fix pack information

Information that is applicable only to the fix pack versions of Tivoli Netcool/OMNIbus are prefaced with a graphic. For example, if a set of instructions is preceded by the graphic Fix Pack 1 , it means that the instructions can only be performed if you installed fix pack 1 of your installed version of Tivoli Netcool/OMNIbus. In the release notes, descriptions of known problems that are prefaced with Fix Pack 1 are solved in fix pack 1, and so on.

Note: Fix packs are distributed separately for the server components and the Web GUI component.

Home directories for the Web GUI and Tivoli Integrated Portal

The Web GUI and the Tivoli Integrated Portal use separate directory structures within the main installation directory. References to those directories use the following conventions:

install_dir
Refers to the directory where the Web GUI and the Tivoli Integrated Portal are installed.

Examples:

`/opt/IBM/tivoli` on UNIX environments.

`C:\IBM\tivoli\` on Windows systems.

webgui-home
Refers to the directory where the Web GUI is installed. This directory is known as the Web GUI home directory.

Examples:

`/opt/IBM/tivoli/netcool/omnibus_webgui` on UNIX environments.

`C:\IBM\tivoli\netcool\omnibus_webgui` on Windows systems.

tip_home_dir
Refers to the directory where the Tivoli Integrated Portal is installed. This directory is known as the Tivoli Integrated Portal home directory.

Examples:

`/opt/IBM/tivoli/tipv2` on UNIX environments.

`C:\IBM\tivoli\tipv2` on Windows systems.

Chapter 1. The Web GUI Administration API (WAAPI)

Read an introduction to WAAPI and how it communicates with the Web GUI server. Also read about the security mechanisms that WAAPI provides.

About WAAPI

WAAPI is an XML-based API that enables remote or local administration of the Web GUI server.

WAAPI enables administration of the server without having to use the graphical user interface of the Web GUI itself. It is particularly useful for carrying out bulk updates of information which would take a long time to achieve using the interactive facilities. For example, adding or modifying a number filters may be more efficient to do using WAAPI rather than the interactive facilities of the Web GUI.

The interface is installed with the Web GUI server enabling administration from the server itself. You can also install the WAAPI client on a remote server and manage the Web GUI from there.

Types of request

An administration command using WAAPI is known as a request. WAAPI allows you to manage many of the facilities available through the interactive interface. For example, you can add, modify, and delete filters.

There are also some features that are available only using WAAPI for specialized administration tasks. For example, you can reload all the system's filters and views.

WAAPI groups all the available requests into the following types:

- User

  Modify the characteristics of any number of users, get a list of users, and remove Web GUI configuration data from users that no longer have active Web GUI roles.
- Views

  Create, modify, delete, and list views.
- Maps

  Create, modify, delete and list maps. In addition, you can add visuals to a map, modify and delete them.
- Resources

  Resources are graphics files that contain items you want to appear on a map. Resources include background images for maps and graphics you want ot use as map objects. You can add remove and list resources.
- Files

  Add and remove directories and files on the Web GUI server.
- Menus

  Create, modify, delete, and list menus to appear on Active Event Lists (AEL).

© Copyright IBM Corp. 2011, 2012

- Tools

  Add, modify, and delete event management tools.
- Prompts

  The Web GUI provides several types of prompt that event management tools can use to obtain information from the user. You can add, modify, delete, and list prompts.
- CGI scripts

  Register, modify, and deregister CGI scripts.
- Filters

  Add, modify, delete, and list filters. In addition, you can default view for a filter.
- Filter collections

  Add, modify, delete, and list filter collections. In addition, you can add filters too and remove them from a filter collection.
- Metrics (gauges)

  Create, modify, delete, and list metrics that the Web GUI displays in the form of gauges.
- Relationships

  Create, modify, delete, and list event relationships.
- Other

  Miscellaneous requests that do not fit in any other category:
  - Resynchronize the Web GUI cache with the ObjectServer.
  - Remove a node from a load balancing cluster.
  - Generate a system status report.
  - Reload all filters and views.

Comparative procedures for modifying the Web GUI

Most of the Web GUI components that you can modify in Tivoli Integrated Portal have an equivalent XML configuration instruction defined in the DTD.

The following examples describe the comparative procedures for modifying a field view in an Active Event List (AEL).

Web GUI procedure

The following example procedure describes how to modify a field view in an AEL through the Web GUI in Tivoli Integrated Portal:

1. Click **Administration** > **Event Management Tools** > **Views**.
2. From the **Views** list, select the view that you want to modify.
3. In the **Available Fields** list, select the field that you want to modify.
4. Change the justification values of “Title” and “Data” to **Left**.
5. Set the column width to **12**.
6. Click **Save**.

Equivalent WAAPI procedure

The following example procedure describes how to modify a field view in an AEL through the WAAPI client:

1. Create an XML command file in accordance with the rules of the WAAPI DTD. For example:

```
<methodCall>
    <method methodName="view.modifyView">
     <view viewName="myview" acl="*">
     <columns>
      <visualEntry fieldName="myfield"
       fieldTitle="myfieldtitle"
       dataJustify="left"
       titleJustify="left"
       columnWidth="12"
       />
     </columns>
     </view>
    </method>
</methodCall>
```

2. Start the WAAPI client and send the command file to the Web GUI server.

Communications between the WAAPI client and the Web GUI server

How the WAAPI client communicates with the Web GUI server.

Characteristics of the communication method

Communication between the WAAPI client and the Web GUI server has the following characteristics:

- Uses a request/response model.
- Is synchronous.
- Requests are in XML format.
- Responses are in text format.
- Uses an HTTP or HTTPS connection between the client and the server.

Communications overview

Whether you use the WAAPI client installed with the Web GUI server or a remote installation of the client, the way it communicates with the server is exactly the same:

1. The administrator creates one or more requests in one or more XML files.
2. The administrator runs the WAAPI client and supplies it with the XML files.
3. WAAPI sends the files to the server over a HTTP connection.

   This connection can use SSL and can use encryption to help maintain the security of the data.
4. The server receives the requests and carries them out.
5. The server returns any output from the requests to the client over the same HTTP connection.
6. The WAAPI client receives the output. How it processes this depends on whether the administrator specified an output file to use when sending the requests.

   If the administrator specified an output file, WAAPI sends the output to that file, creating it if necessary. Otherwise WAAPI sends the output to the screen where the administrator ran the client.

Security

The WAAPI client has a number of security features that help to protect the integrity of the data it exchanges with the Web GUI client.

WAAPI provides three ways you can use to protect the data it exchanges with the server:

- Securing the connection to the server
- Password encryption
- Protecting the WAAPI properties file

Secure Connections to the Web GUI server

In place of an unprotected HTTP connection, you can set up a secure connection with the Web GUI server using SSL. You can set up this connection in any of the following ways:

- Server-only authentication without FIPS 140-2
- Server and client authentication without FIPS 140-2
- Server-only authentication with FIPS 140-2
- Server and client authentication with FIPS 140-2

Password encryption

Independently of any secure connection you might use, WAAPI provides the means for encrypting the passwords that it uses. An unprotected HTTP connection can use AES password encryption. When using a secure connection, you can specify AES or FIPS 140-2 encryption. When the connection uses FIPS 140-2, only FIPS 140-2 password encryption is available.

Protecting the WAAPI properties file

The WAAPI properties file (`waapi.init`) contains a number of sensitive items of data. For example, it often holds the username and password of the administrative user on the server that runs WAAPI requests. It is important that this data is kept away from unauthorized users and is available only to administrators. So you can use the access control mechanisms of the operating system to limit access to the file.

Related reference:

Appendix C, “WAAPI security,” on page 99
WAAPI has a number of security features that help to ensure secure communication with the Web GUI server.

Chapter 2. Using WAAPI

How to set up the WAAPI properties file for your environment and to use the WAAPI client to send requests to the Web GUI server.

Setting up the WAAPI properties file

Use the WAAPI properties (`waapi.init`) to match your environment.

Procedure

Set the properties in the WAAPI initialization file to match your environment. Set values for those properties whose value changes rarely such as:

- **`waapi.host`**
- **`waapi.port`**
- **`waapi.user`**
- **`waapi.password`**
- The secure connection and encryption properties

This minimizes the number of options that you need to enter on the command line.

Note: If you set the **`waapi.user`** and **`waapi.password`** properties, make sure that the properties file is protected against access for users other than authorized administrators.
If you manage several Web GUI servers from one location, you can have multiple properties files, one for each server. This enables you to tailor the properties for each server.
You can also use the properties file to hold default values for properties. You can always override any property setting using a command line option.

Related reference:

Appendix A, "WAAPI properties and command-line options," on page 93
Use this information to learn about the properties and command-line options of the WAAPI client.

Sending WAAPI requests to the server

To send requests to the Web GUI server, prepare the WAAPI command file that contains the requests and run the WAAPI client to send it to the server.

Procedure

1. To prepare the WAAPI command file:
   a. Create an 8-bit Unicode Transformation format (UTF) text file with a `.xml` suffix.
   b. Add the requests to the file

      Tip: Use the sample WAAPI files provided with the client as templates for your command file.
   c. Save the file using UTF-8 to a suitable directory.
2. To run the client and send your file to the server:

© Copyright IBM Corp. 2011, 2012

a. Enter the following command, dependent on the operating system you use:

- UNIX Linux *webgui-home*/waapi/bin/runwaapi *options* -file *waapi_command_file*
- Windows *webgui-home*\waapi\bin\runwaapi.cmd *options* -file *waapi_command_file*

Replace:

web_gui_home_dir
With the installation directory of the Web GUI. If you are running the client on a remote system, specify the location where you installed the client.

options
With any additional command-line options that you require. Commonly used options are described in the following table.

Table 1. Frequently used WAAPI command-line options

Function	Option
Redirect output to a file	-outfile *filepath* Replace *filepath* with the path of the file to receive output from the WAAPI command file.
Specify a user name and password to run the command file under	-user *username* -password *password* Replace *username* with the name of the account to use, and *password* with the password for that account. **Note:** If you use these options, clear the screen and the command history to ensure that the credentials remain secure.
Use an alternative properties file	-props *propsfile* Replace *propsfile* with the path name of the WAAPI properties file to use.

waapi_command_file
With the name of your WAAPI command file.

For example, on a Windows system:

```
c:\ibm\tivoli\netcool\omnibus_webgui\waapi\bin\runwaapi.cmd
-file newFilters.xml
```

Chapter 3. WAAPI requests

A WAAPI request is an XML document that contains instructions to administer the Web GUI server.

You can administer the following items using WAAPI requests:

- "User requests" on page 11
- "View requests" on page 19
- "Map requests" on page 26
- "Resource requests" on page 46
- "File requests" on page 48
- "Menu requests" on page 51
- "Tool requests" on page 55
- "Prompt requests" on page 64
- "CGI requests" on page 66
- "Filter requests" on page 68
- "Filter collection requests" on page 74
- "Metric requests" on page 79
- "Relationship requests" on page 86
- "Other requests" on page 90

Note: The names of some attributes are long. In the following syntax descriptions, these long names are divided over 2 or more lines. However, in the XML you produce, enter each attribute name as a single character sequence without any line breaks.

Structure of a WAAPI request

A WAAPI request is an XML document that contains an optional XML declaration followed by the `<methodCall>` root element. A request has a number of additional characteristics that you need to bear in mind.

- "The XML declaration"
- "The <methodCall> root element"
- "Basic form of the root element" on page 8
- "The root element for tool, prompt, metric, and relationship requests" on page 8
- "Content of the root element" on page 8

The XML declaration

Optionally, you can begin a WAAPI request with an XML declaration. For this release of the Web GUI the declaration is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

The <methodCall> root element

The root element holds the content of the request. For tool, prompt, and metric requests, the root element also defines a namespace.

© Copyright IBM Corp. 2011, 2012

Basic form of the root element

The basic form of the root element is as follows:

```
<methodCall>

</methodCall>
```

The root element for tool, prompt, metric, and relationship requests

For tool, prompt, metric, and relationship requests, the form of the root element is as follows:

```
<methodCall xmlns:type=namespace-url>

</methodCall>
```

Where *type* is the type of request (tool, prompt, metric, or relationship) and *namespace-url* is the fully qualified URL for the namespace. Only the tool, prompt, metric, and relationship requests require a namespace URL.

The namespace URLs for prompt, tool, metric, and relationship requests are described in the following table.

Table 2. Namespace URLs

Type of request	Namespace URL
Tool	"http://www.ibm.com/tivoli/netcool/webtop/tools/2.1"
Prompt	"http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2"
Metric	"http://www.ibm.com/tivoli/netcool/webtop/metrics/7.3.1"
Relationship	"http://www.ibm.com/tivoli/netcool/webtop/relationships/7.4"

Content of the root element

The root element, whatever form it takes, contains one or more `<method>` elements. Each of these elements contains a request to manipulate an item of Web GUI data. Later sections set out the format and content of the `<method>` element for each type of data that WAAPI can operate on. The `<method>` element contains a `methodName` attribute that defines the type of data item and the operation to perform on that data. In the `<method>` element are child elements that define the item of data.

For example, when the `methodCall` attribute has the value `view.createView` the `<method>` element contains one or more `<view>` elements each of which defines the characteristics of a view to add to the collection of Web GUI views.

The root element can contain method calls for any mix of Web GUI data items. When the element contains any combination of tool, prompt, metric, and relationship requests, include each of the corresponding xmlns attributes in the `<methodCall>` element.

For example, if a `<methodCall>` element contains `<method>` elements for views, tools, and prompts, specify the `<methodCall>` element as follows:

```
<methodCall xmlns="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1"
 xmlns="http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2" >

 <!-- <method> elements appear here -->

</methodCall>
```

Characteristics of WAAPI XML documents

XML documents that contain WAAPI requests have a number of characteristics that you need to bear in mind:

- "Document Type definition (DTD)"
- "The order of XML elements"
- "Case sensitivity" on page 10
- "Content and value restrictions" on page 10
- "Comments" on page 10

Document Type definition (DTD)

Each WAAPI request must be well-formed and is validated against the document type definition (DTD) by the WAAPI client. The DTD defines a set of rules for the syntax of elements that appear in the request.

The DTD defines the statements that you can put in the XML request, the order in which elements must appear, which elements can be nested, which elements have attributes, and so forth. The WAAPI DTD is in the following location:

webgui-home/waapi/etc/waapi.dtd

The order of XML elements

In general, the WAAPI DTD is order tolerant. Providing that the request files are well formed, the DTD allows you to parse the majority of child statements. However, some elements required that you put child elements in a particular order.

An example of an order-tolerant element is <supermenu>. The <supermenu> element can contain three types of child element: <tool>, <separator>, and <menu>. These elements can appear in any order within the <supermenu> element hierarchy. The order in which elements are placed in the appearance of the AEL tool menu that is created by these instructions.

On the other hand, the child elements of the <method> and <view> elements must appear in the correct order. The following table defines the order in which child elements for the <method> and <view> must appear.

Table 3. Child element order

Element	Child element order
method	1. user 2. view 3. filter 4. map 5. resources 6. supermenu 7. tools 8. cgi 9. file 10. tool:tool 11. prompt:prompt 12. filterCollection 13. metric:metric 14. relationship:relationship
view	1. columns 2. sorting

Case sensitivity

In XML documents, element and attribute names are case sensitive. Use the same case as specified in the method definitions. In addition, the content of some elements and the value of some attributes are also case sensitive. The descriptions of element content and attribute values in the method definitions includes information on case sensitivity. In particular be sure to specify enumerated values exactly as they appear in the method definition.

Content and value restrictions

There are restrictions on the content of some elements and the value of some attributes. For example, there may be a limit on the number of characters that an attribute value can have. The description of elements and attributes in each method definition define any content restrictions that apply.

Comments

You can include comments in the WAAPI request using the standard XML syntax. Begin the comment with the syntax `<!--` and terminate the comment with the syntax `-->`. Put the comment between the beginning and terminating syntax. A comment can be on a single line or multiple lines.

For example:

```
<!-- This is a comment -->

<!--
 This is a comment that has more
 than one line.
-->
```

Sample Requests

There are a wide range of sample requests supplied with the Web GUI.

The installation of the Web GUI server includes set of example requests. You can use these as models for your requests. The examples are in the following directory:

webgui-home/waapi/etc/samples

Here, *webgui-home* is the installation directory of the Web GUI. For example, ibm/tivoli/netcool/omnibus_webgui.

User requests

User requests operate on Web GUI users. There are functions to modify a user, get a list of users, and remove configuration information for users that do not have Web GUI user privileges. WAAPI provides three methods for working on users defined in the system: maintaining users, modifying users, and getting a list of users.

Maintain users

The format of the <method> element for maintaining users is:

```
<method methodName="user.maintainUsers" />
```

Use this method to remove Web GUI configuration data from all users that no longer have the ncw_user or ncw_admin roles. The configuration data includes:

- Preferences
- Filter definitions
- View definitions

Example

```
<methodCall>
  <method methodName="user.maintainUsers" />
</methodCall>
```

Modify a user

The format of the <method> element for modifying a user is:

```
<method methodName="user.modifyUser">
```

Use this method call to change any number of the following characteristics of a user:

- Default filter
- Home page
- AEL characteristics, including:
  - Access to the AEL
  - Default and minimum refresh time
  - Items to display in the AEL

The <method> element contains one or more <user> elements each of which identifies a user and the characteristics of the user that are to be modified. Include only attributes of the <user> that correspond to the characteristics you want to change. When you omit an attribute the corresponding characteristic is unchanged.

<user>

The <user> element defines the characteristics of a user and has the following attributes:

Table 4. Attributes of the <user> element

Attribute name	Required or optional	Description
name	Required	Identifies the user to modify. Value: The username of a Web GUI user. Default value: None.
filter	Optional	Defines the event severity levels that appear in the user's AEL. Value: String Default value: None.
homepage	Optional	The URL of the user's home page, relative to the context root of the Web GUI. Value: The name and location of the home page. Default value: /index.html
ael_user	Optional	Specifies whether the user can access the Active Event List (AEL). Value: true or false Default value: true
ael_user_properties _acknowledge_font_color	Optional	Defines the font color for acknowledged events. Value: String Default value: White
ael_user_properties_allow _select	Optional	Specifies whether the user can set their own preferences in the ACL. Value: true or false Default value: true
ael_user_properties_allow _custom_refresh	Optional	Specifies whether the user can refresh the AEL display. The attribute has one of the following values: Value: true or false Default value: false
ael_user_properties_refresh _time	Optional	Specifies the refresh time (in seconds) of the AEL for this user. Value: Integer Default value: 60

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties _minimum_refresh_time	Optional	Specifies the minimum refresh time (in seconds) that the user can specify. The value of this attribute must be less than or equal to the value of `ael_user_properties_refresh_ time`. Value: Integer. Default value: `30`
ael_user_properties_show _colors	Optional	Specifies whether colors are used in the user's AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_show_info	Optional	Specifies whether the **Alerts** menu of the AEL includes the option to display the Information window. Value: `true` or `false` Default value: `true`
ael_user_properties_show _journal	Optional	Specifies whether the Information window for an alert includes the **Journal** tab. Value: `true` or `false` Default value: `true`
ael_user_properties_show _details	Optional	Specifies whether the Information window for an alert includes the **Details** tab. Value: `true` or `false` Default value: `true`
ael_user_properties_show _menubar	Optional	Specifies whether the AEL portlet includes a menu bar. Value: `true` or `false` Default value: `false`
ael_user_properties _showgraphicconversions	Optional	Specifies whether the AEL displays severity icons. Value: `true` or `false` Default value: `false`
ael_user_properties_show _preferences	Optional	Specifies whether the user can access the Preferences window from the AEL **Edit** menu or the Preferences icon on the toolbar. Value: `true` or `false` Default value: `true`

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties_monitor _show_number	Optional	Specifies whether the total number of events appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `true`
ael_user-properties_monitor _show_highest	Optional	Specifies whether the highest severity level appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor _show_highest_color	Optional	Specifies whether the color of the highest severity level appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor _show_lowest	Optional	Specifies whether the lowest severity level appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor _show_lowest_color	Optional	Specifies whether the color of the lowest severity level appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `false`
ael_user_properties_monitor _show_border	Optional	Specifies whether the border around monitor boxes in the Event Dashboard, or in an AEL displayed in monitor box style using SmartPage commands, has the same color as the maximum severity displayed in the box or AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor _show_metric	Optional	Specifies whether the metric appears on monitor boxes in the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor_ distribution_meter	Optional	Specifies the type of distribution meter to use for representing alerts on the user's AEL monitor boxes. Value: `histogram`, `lavalamp`, or `none`. Default value: `histogram`

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties_flash _time	Optional	Specifies the flash time in milliseconds for alerts in the AEL. This attribute is effective only when the value of ael_user_properties_flash_enabled is `true`. Value: Integer Default value: `400`
ael_user_properties_flash _brightness	Optional	Specifies the flash brightness of alerts in the AEL. This attribute is effective only when the value of ael_user_properties_flash_enabled is `true`. Value: Integer Default value: `0`
ael_user_properties_flash _enabled	Optional	Specifies whether unacknowledged events in the AEL flash. Value: `true` or `false` Default value: false
ael_user_properties_show _summarybar	Optional	Specifies whether the summary bar appears at the foot of the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_show _toolbar	Optional	Specifies whether the toolbar appears above the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_monitor _font_name	Optional	Specifies the name of the font to use for text on monitor boxes in the AEL. Value: String Default value: `Dialog`
ael_user_properties_monitor _font_size	Optional	Specifies the size of the font used on monitor boxes in the AEL. Value: Integer Default value: 12
ael_user_properties_timeformat	Optional	Specifies the format for the date and time as it appears on the AEL. Value: `short`, `long`, or a date and time format. Default value: `short`
ael_user_properties_eventlist _font_name	Optional	Specifies the name of the font to use on the AEL. Value: String Default value: `Dialog`

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties_eventlist _font_size	Optional	Specifies the size of the font to use on the AEL. Value: Integer Default value: `12`
ael_user_properties_eventlist _width	Optional	Specifies the total width, in pixels, of the AEL. Value: Integer Default value: `600`
ael_user_properties_eventlist _height	Optional	Specifies the total height, in pixels, of the AEL. Value: Integer Default value: `450`
ael_user_properties_notify _enabled	Optional	Specifies whether notifications (for example, sounds) occur when events are added to or modified on the AEL. Value: `true` or `false` Default value: `false`
ael_user_properties_notify _when_iconized	Optional	When the user has iconized the AEL, specifies whether to redisplay it when a new event arrives. Value: `true` or `false` Default value: `true`
ael_user_properties_notify _always	Optional	Specifies whether the user receives a notification of the arrival of an event irrespective of whether the AEL is open, closed, or iconized. Value: `true` or `false` Default value: `true`
ael_user_properties_notify _insert	Optional	Specifies whether the user receives a notification when an event is added to the AEL. Value: `true` or `false` Default value: `false`
ael_user_properties_notify _delete	Optional	Specifies whether the user receives a notification when an event is removed from the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_notify _update	Optional	Specifies whether the user receives a notification when any event in the AEL is updated. Value: `true` or `false` Default value: `false`

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties_notify _play_sound	Optional	Specifies whether a notification to use includes playing a sound. Value: `true` or `false` Default value: `true`
ael_user_properties_notify _sound_url	Optional	Specifies the URL of a sound to play as a notification. Value: String Default value: none
ael_user_properties_notify _flash_icon	Optional	Specifies whether the AEL icon flashes as part of a notification. Value: `true` or `false` Default value: `true`
ael_user_properties_notify _open_window	Optional	Specifies whether a window opens as part of a notification. Value: `true` or `false` Default value: `false`
ael_user_properties_notify _open_url	Optional	Specifies whether to open a URL as part of a notification. Value: `true` or `false` Default value: `false`
ael_user_properties_notify_url	Optional	Specifies the URL to open when the value of ael_user_properties_notify_open_url is `true`. Value: String Default value: none
ael_user_properties_notify _url_target	Optional	Specifies the target in the browser where the URL specified in ael_user_notify_url opens. Where frames have been defined on the HTML you can specify the target as the name of a frame (for example, UpperFrame). Value: `_blank`, the name of a frame Default value: none
ael_user_properties_monitor _num_cols	Optional	Specifies the number of columns of monitor box applets to display on the AEL. Value: Integer Default value: `4`
ael_user_properties_allow _journal_edit	Optional	Specifies whether the user can edit the journals associated with the AEL. Value: `true` or `false` Default value: `true`

Table 4. Attributes of the <user> element (continued)

Attribute name	Required or optional	Description
ael_user_properties_allow _filter_builder_access	Optional	Specifies whether the user can access the Filter Builder from the AEL. Value: `true` or `false` Default value: `false`
ale_user_properties_allow _view_builder_access	Optional	Specifies whether the user can access the View Builder from the AEL. Value: `true` or `false` Default value: `true`
ael_user_properties_allow _views_and_filters_use	Optional	Specifies whether the user can select pre-defined filters and views in the AEL. Value: `true` or `false` Default value: `true`
map_editor_user_properties _show_grid	Optional	Specifies whether a grid appears in the Map Editor. Value: `true` or `false` Default value: `true`
map_editor_user_properties _snap_to_grid	Optional	Specifies whether objects in the Map Editor snap to the grid. Value: `true` or `false` Default value: `false`
map_editor_user_properties _grid_size	Optional	Specifies the size of the grid in the Map Editor. Value: Integer Default value: `5`
map_editor_user_properties _editor_width	Optional	Specifies the total width, in pixels, of the display area in the Map Editor. Value: Integer Default value: `600`
map_editor_user_properties _editor_height	Optional	Specifies the total height, in pixels, of the display are in the Map Editor. Value: Integer Default value: `400`

Example

This example modifies the user named `user1`, and defines the following characteristics:

- The default filter shows all severities.
- The following AEL characteristics are modified:
  - Refresh time is 90 seconds
  - The minimum refresh is 70 seconds
  - The user can access the AEL Preferences window

- The user cannot force a refresh of the AEL
- The user can set their own preferences
- The AEL Information window includes the **Details** and **Journal** tabs
- The **Alerts** menu includes an option to open the Information window.

```
<methodCall>
  <method methodName="user.modifyUser">
    <user name="user1"
      filter ="Severity>=0"
      ael_user_properties_refresh_time="90"
      ael_user_properties_minimum_refresh_time="70"
      ael_user_properties_show_preferences ="true"
      ael_user_properties_allow_custom_refresh = "false"
      ael_user_properties_allow_select  = "true"
      ael_user_properties_show_details ="true"
      ael_user_properties_show_info  ="false"
      ael_user_properties_show_journal = "true">
    </user>
  </method>
</methodCall>
```

Get a list of users

The format of the <method> element for getting a list of users is:

```
<method methodName="user.getList" />
```

Use this method to obtain a list containing the login user names of all users defined in the Web GUI that have either the ncw_user or ncw_admin roles.

Example

```
<methodCall>
  <method methodName="user.getList" />
</methodCall>
```

View requests

View requests operate on Web GUI views. There are functions to create, modify, and delete views. In addition you can obtain a list of the views defined on the Web GUI server. WAAPI provides five methods for working with views: creating views, creating or replacing views, modifying views, deleting views, and getting a list of views.

Create a view

The format of the <method> element for creating a view is:

```
<method methodName="view.create">
```

Use this method to create a new view for use on the AEL, Event Viewer, LEL, Table View, and Event Dashboard. The <method> element contains one or more <view> elements, each of which defines the characteristics of a new view. The <view> element can contain up to one of each of the <columns>, <sorting>, and <grouping> elements.

- "<view>" on page 20
- "<columns>" on page 20
- "<visualEntry>" on page 20
- "<sorting>" on page 21
- "<sortColumn>" on page 21
- "<grouping>" on page 22

- "<groupColumn" on page 22
- "Example user view" on page 23

<view>

The <view> element defines a view and has the following attributes:

Table 5. Attributes of the <view> element

Attribute name	Required or optional	Description
viewName	Required	Provides a unique name for a view. Value: String Default value: None
datasource	Optional	The name of the data source that provides events for the view. To specify multiple data sources use a comma-separated list. Value: String Default value: NCOMS
user	Optional	For user views, this attribute identifies the users that the view is associated with. The value of the attribute is a comma-separated list of User IDs. Value: List of User IDs Default value: None
type	Optional	The type of view. Value: global, system, group, or user Default value: None
relationshipName	Optional	The name of an event relationship to use with the view that is applied to the Event Viewer. An event relationship organizes the Event Viewer by the relationships between them. For example, by their root cause and symptom. Value: String Default value: None

<columns>

The <columns> element is a child the <view> element and can occur once or not at all. If present, the element contains any number of <visualEntry> elements.

<visualEntry>

The<visualEntry> element is a child of the <columns> element that defines the appearance of fields in the view. The element has the following attributes:

Table 6. Attributes of the <visualEntry> element

Attribute name	Required or optional	Description
fieldName	Required	The name of a field to include in the view. This is analogous to an entry from the **Available fields** list in the View Builder. Value: The name of field from a ObjectServer table. Default value: None
fieldTitle	Required	The name to use for the column title in the view for this field. Value: String Default value: None
dataJustify	Required	Specifies how to justify the text in the column. Value: `centre`, `left`, or `right`. Default value: `left`
titleJustify	Required	Specifies how to justify the column title. Value: `centre`, `left`, or `right` Default value: `left`
columnWidth	Rquired	The width of the column, in pixels. Value: Integer Default value: `25`
columnLocked	Optional	Specifies whether the position of a column is locked or whether the user can rearrange the order of the column. Value: `true` or `false` Default value: `false`
datasource	Optional	The name of the datasource that provides data for the field. Value: Name of a data source Default value: `NCOMS`

<sorting>

The `<sorting>` element is a child of the `<view>` element and can occur once or not at all. If present, the element contains any number of `<sortColumn>` elements that define the sorting order for events in the view.

<sortColumn>

The `<sortColumn>` element defines a field to use when sorting the list events in a view. When there are multiple `<sortColumn>` the entries in the view are sorted in the order that the `<sortColumn>` elements appear.

The element has the following attributes:

Table 7. Attributes of the <sortColumn> element

Attribute name	Required or optional	Description
fieldName	Required	The name of a field in the view. Value: The name of a field. Default value: None.
order	Required	The order to sort the column entries. Value: `asc` (for ascending order) or `desc` (for descending order). Default value: `asc`
datasource	Optional	The name of the data source that provides the named field. Value: The name of a data source. Default value: `NCOMS`

<grouping>

The `<grouping>` element is a child of the `<view>` element and can occur once or not at all. If present, the element contains any number of `<groupColumn>` elements that define how events are grouped in the Event Viewer when using this view.

<groupColumn

The `<groupColumn>` element defines a field to use when grouping the list events in a view used by the Event Viewer. When there are multiple `<groupColumn>` elements, the entries in the view are grouped in the order that the `<groupColumn>` elements appear.

The element has the following attributes:

Table 8. Attributes of the <groupColumn> element

Attribute name	Required or optional	Description
fieldName	Required	The name of a field in the view. This is analogous to an entry from the **Available fields** list in the View Builder. Value: The name of a field. Default value: None.

Notes:

- The WAAPI DTD allows any number of `<groupColumn>` elements. However, the maximum number of levels of grouping that you can use is set by the **`columngrouping.maximum.columns`** property in the Web GUI initialization file (`server.init`). By default, the value of that property is 3 but the administrator can set a different value.

  If you specify more levels than are defined by this property, the server returns an error.
- The **`columngrouping.allowedcolumns`** property in `server.init` defines the names of the fields that you can use for grouping. The default value of that property is:

```
Acknowledged,AlertGroup,Class,Customer,Location,Node,NodeAlias,
NmosCauseType,NmosManagedStatus,Severity,Service
```

The administrator can change that value. If you specify a column that is not in the list for that property, the server returns an error.

Example user view

The following example creates a user view named `SeveritySummary` that has the following characteristics:

- The view includes the following fields formatted as shown:

Table 9. Create user view example: Fields and their formatting

Field	Field title	Data justify	Title justify	Column width
Severity	Sev	Center	Center	5
Acknowledged	Ack	Center	Center	3
Node	Node	Left	Left	12
AlertGroup	Alert Group	Left	Left	10
Summary	Summary	Left	Left	40
LastOccurrence	Last Occurrence	Left	Left	14

- The columns for the Severity and Acknowledged columns are locked.
- The view uses the default data source.
- The view sorts entries in descending order of Severity.
- The view is available to the ncoadmin and tipadmin users.
- The view groups entries by Node

```
<methodCall>
 <method methodName="createView">
  <view viewName="SeveritySummary"
   user="ncoadmin,tipadmin"
   type="user">
    <columns>
     <visualEntry fieldName="Severity"
      fieldTitle="Sev"
      dataJustify="centre"
      titleJustify="center"
      columnWidth="5"
      columnLocked="true" />
     <visualEntry fieldname="Acknowledged"
      fieldTitle="Ack"
      datajustify="centre"
      titleJustify="center"
      columnWidth="3"
      columnLocked="true" />
     <visualEntry fieldName="Node"
      fieldTitle="Node"
      dataJustify="left"
      titleJustify="left"
      columnWidth="12" />
     <visualEntry fieldName="AlertGroup"
      fieldTitle="Alert Group"
      dataJustify="left"
      titleJustify=left"
      columnWidth="10" />
     <visualEntry fieldName="Summary"
      fieldTitle="Summary"
      dataJustify="left"
      titleJustify="left"
```

```
      columnWidth="40" />
     <visualEntry fieldName="LastOccurrence"
      fieldTitle="Last Occurrence"
      dataJustify="left"
      titleJustify="left"
      columnWidth="14" />
    </columns>
    <sorting>
     <sortColumn fieldName="Severity" order="desc" />
    </sorting>
    <grouping>
     <groupColumn fieldName="Node" />
    </grouping>
  </view>
 </method>
</methodCall>
```

Create or replace a view

The format of the <method> element for creating or replacing a view is:

```
<method methodName="view.createOrReplaceView">
```

Use this to replace an existing view or to create a new one if it does not already exist. The <method> element contains one or more <view> elements, each of which defines the characteristics of a view. Each <view> element can contain up to one <columns> element, up to one <sorting> element, and up to one <grouping> element. If present, the <columns> element contains any number of <visualEntry> elements and, if present, each <sorting> element contains any number of <sortColumn> elements. If present, the <grouping> element can contain any number of <groupColumn> elements.

For more information about the <view> element and its subelements, see "<view>" on page 20.

This example creates or replaces a view named OccurrenceSummary that summarizes the last occurrence of alerts. It contains four columns: Severity, Node, Summary, and LastOccurrence. These columns are formatted in the same way as the corresponding columns in the example of creating a view.

```
<methodCall>
 <method methodName="view.createOrReplaceView">
  <view viewName="OccurrenceSummary"
   user="ncoadmin,tipadmin"
   type="user">
    <columns>
     <visualEntry fieldName="Severity"
      fieldTitle="Sev"
      dataJustify="centre"
      titleJustify="center"
      columnWidth="5"
      columnLocked="true" />
     <visualEntry fieldName="Node"
      fieldTitle="Node"
      dataJustify="left"
      titleJustify="left"
      columnWidth="12" />
     <visualEntry fieldName="Summary"
      fieldTitle="Summary"
      dataJustify="left"
      titleJustify="left"
      columnWidth="40" />
     <visualEntry fieldName="LastOccurrence"
      fieldTitle="Last Occurrence"
      dataJustify="left"
```

```
      titleJustify="left"
      columnWidth="14" />
    </columns>
    <sorting>
     <sortColumn fieldName="LastOccurrence" order="desc" />
    </sorting>
  </view>
 </method>
</methodCall>
```

Modify a view

The format of the <method> element for modifying a view is:

```
<method methodName="view.modifyView">
```

Use this method to modify the characteristics of an existing view. Include any or all of the <columns>, <sorting>, and <grouping> elements, with their child elements, as required. For example, to change the sorting method of the view, include the <sorting> element with the <sortColumn> elements necessary to define the sorting method.

For more information about the <view> element and its subelements, see "<view>" on page 20.

Example

This example modifies the view named view1 and makes the following changes:

- The view includes two fields:

  Node

  Serial

- The view sorts entries on ascending order of the Severity field.

```
<methodCall>
 <method methodName="view.modifyView">
  <view viewName="view1">
   <columns>
    <visualEntry fieldName="Node"
     fieldTitle="Node"
     dataJustify="left"
     titleJustify="left"
     columnWidth="18" />
    <visualEntry fieldName="Serial"
     fieldTitle="Serial"
     dataJustify="left"
     titleJustify="left"
     columnWidth="12" />
   </columns>
   <sorting>
    <sortColumn fieldName="Severity" order="asc" />
   </sorting>
  </view>
 </method>
</methodCall>
```

Delete a view

The format of the <method> element for deleting a view is:

```
<method methodName="view.delete">
```

The <method> element contains one or more <view> elements each of which identifies a view to delete. In the view element, include only the viewName attribute.

For more information about the <view> element and its subelements, see "<view>" on page 20.

Example

This example deletes the view named viewsample2.

```
<methodCall>
 <method methodName="view.deleteView">
  <view viewName="viewsample2">
  </view>
 </method>
</methodCall>
```

Get a list of views

The format of the <method> element for getting a list of views is:

```
<method methodName="view.getList">
```

The method returns a list of the names of all views defined in the Web GUI. There is a separate section in the listing for each type of view.

This method does not return the attributes of the lists, only the names.

Example

```
<methodCall>
 <method methodName="view.getList">
 </method>
</methodCall>
```

Map requests

Maps provide a visual means of viewing events and their locations. There are functions to create, modify, and delete maps and map visuals. In addition, you can obtain a list of the maps defined on the server. WAAPI provides five methods for working with maps: Creating maps, creating or replacing maps, modifying maps, deleting maps, and getting a list of maps. WAAPI provides four methods for working with map visuals: Adding map visuals, adding or replacing map visuals, modifying maps visuals, and deleting map visuals.

Create a map

The format of the <method> element for creating a map is:

```
<method methodName="map.createMap">
```

Use this method to create a new map. The <method> element contains one or more <map> elements each of which defines the characteristics of a new map. The <map> element contains any number of the <text>, <button>, <monitor>, <line>, and <icon> elements.

- "<map>"
- "<text>"
- "<button>" on page 30
- "<monitor>" on page 33
- "<icon>" on page 36
- "<line>" on page 39
- "Example" on page 42

<map>

The <map> element defines a map and has the following attributes:

Table 10. Attributes of the <map> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a map. Value: String Default value: None
bgImage	Optional	The path of an image to use as the background for the map. Value: String Default value: None
bgColor	Optional	The name of a color to use as the background for the map. Value: String Default value: None
h	Optional	The height of the map in pixels. Value: Integer Default value: None
w	Optional	The width of the map in pixels. Value: Integer Default value: None

<text>

The <text> element is a child of the <map> element and defines the characteristics of a text label on a map. The element can occur any number of times and, if present, has the following attributes:

Table 11. Attributes of the <text> element

Attribute name	Required or optional	Description
filter	Optional	Defines a filter to associate with the field. Value: String Default value: None
filterType	Optional	The type of the filter defined in the filter attribute. Value: `global` or `system` Default value: None
name	Required	The name of the text label. Value: String Default value: None
label	Optional	The text to appear on the label. Value: String Default value: None
datasource	Optional	The name of the data source to associate with the label. To specify more than one data source, use a comma-separated list. Value: String Default value: `NCOMS`
x	Optional	The horizontal position of the label in pixels from the left of the map Value: Integer Default value: None
y	Optional	The vertical position of the label in pixels from the bottom of the map. Value: Integer Default value: None
translucency	Optional	The level of translucency for the label. Value: Integer between 0 and 100 (inclusive). Default value: None
rotate	Optional	The angle in degrees to rotate the label. Value: Integer between 0 and 360 (inclusive) Default value: None
show_shadow	Optional	Specifies whether the label appears on the map with a shadow. Value: `true` or `false` Default value: `false`

Table 11. Attributes of the <text> element (continued)

Attribute name	Required or optional	Description
font	Optional	The name of the font to use for the label. Value: String Default value: None
size	Optional	The point size of the font to use for the label. Value: Integer Default value: None
justify	Optional	Defines alignment of the text in the label. Value: `center`, `left`, or `right` Default value: None
style	Optional	The text style (for example, bold) for the label. Value: String Default value: None
color	Optional	The name of the color to use for the text. Value: String Default value: None
action	Optional	The action that takes place when the user clicks on the text label. Value: String. Possible values for opening event lists are as follows: • `"Open URL"` or `"go"` to open a URL. The URL is specified by the url attribute. • `"Active Event List (AEL)"` or `"ael"` to open an Active Event List. • `"Lightweight Event List (LEL)"` to open a Lightweight Event List. • `"Event Table (Table View)"` to open a Table View. • `"Event Viewer"` to open an Event Viewer. • `"Update Event List (using wires)"` to send a NodeClickedOn event using Tivoli Integrated Portal wires. For this option, ensure that a system or custom wire to an Event Viewer or AEL is configured. Default value: None
url	Optional	The uniform resource locator that is the action target for the text label. Value: String Default value: None

Table 11. Attributes of the <text> element (continued)

Attribute name	Required or optional	Description
target	Optional	The target window in the browser for output from the value of the url attribute. When the attribute is not present, output replaces the content of the current browser window. Value: String Default value: `_blank`
flash	Optional	Defines whether the text label flashes. Value: `true` or `false` Default value: `false`

<button>

The `<button>` element is a child of the `<map>` element and defines the characteristics of a button on a map. The element can occur any number of times and, if present, has the following attributes:

Table 12. Attributes of the <button> element

Attribute name	Required or optional	Description
filter	Optional	Defines a filter to associate with the button. Value: String Default value: None
filterType	Optional	The type of the filter defined in the filter attribute. Value: `global` or `system` Default value: None
name	Required	The name of the button. Value: String Default value: None
label	Optional	The text to appear on the button. Value: String Default value: None
datasource	Optional	The name of the data source to associate with the button. To specify more than one data source, use a comma-separated list. Value: String Default value: `NCOMS`
x	Optional	The horizontal position of the button in pixels from the left of the map Value: Integer Default value: None

Table 12. Attributes of the <button> element (continued)

Attribute name	Required or optional	Description
y	Optional	The vertical position of the button in pixels from the bottom of the map. Value: Integer Default value: None
translucency	Optional	The level of translucency for the button. Value: Integer between 0 and 100 (inclusive). Default value: None
show_shadow	Optional	Specifies whether the button appears on the map with a shadow. Value: `true` or `false` Default value: `false`
font	Optional	The name of the font to use for the button text. Value: String Default value: None
size	Optional	The point size of the font to use for the button text. Value: Integer Default value: None
font_color	Optional	The name of the color to use for the button text. Value: String Default value: None
style	Optional	The text style (for example, bold) for the button text. Value: String Default value: None
color	Optional	The name of the color to use for the button. Value: String Default value: None

Table 12. Attributes of the <button> element (continued)

Attribute name	Required or optional	Description
action	Optional	Value: String. Possible values for opening event lists are as follows: • `"Open URL"` or `"go"` to open a URL. The URL is specified by the url attribute. • `"Active Event List (AEL)"` or `"ael"` to open an Active Event List. • `"Lightweight Event List (LEL)"` to open a Lightweight Event List. • `"Event Table (Table View)"` to open a Table View. • `"Event Viewer"` to open an Event Viewer. • `"Update Event List (using wires)"` to send a NodeClickedOn event using Tivoli Integrated Portal wires. For this option, ensure that a system or custom wire to an Event Viewer or AEL is configured. Default value: None
url	Optional	The uniform resource locator that is the action target for the button. Value: String Default value: None
target	Optional	The target window in the browser for output from the value of the url attribute. When the attribute is not present, output replaces the content of the current browser window. Value: String Default value: `_blank`
flash	Optional	Defines whether the button flashes. Value: `true` or `false` Default value: `false`
w	Optional	The width of the button in pixels. Value: Integer Default value: None
h	Optional	The height of the button in pixels. Value: Integer Default value: None
type	Optional	The type of button, for example a rectangle. Value: `rectangle`, `rounded`, or `ellipse` Default value: `rectangle`

Table 12. Attributes of the <button> element (continued)

Attribute name	Required or optional	Description
arc_diameter	Optional	The arc diameter of the button, in degrees. Value: Integer Default value: None
transparent	Optional	Defines whether the button is transparent. Value: `true` or `false` Default value: `false`
legend	Optional	The legend for the button. Value: String Default value: None

<monitor>

The `<monitor>` element is a child of the `<map>` element and defines the characteristics of a monitor box on a map. The element can occur any number of times and, if present, has the following attributes:

Table 13. Attributes of the <monitor> element

Attribute name	Required or optional	Description
filter	Optional	Defines a filter to associate with the monitor box. Value: String Default value: None
filterType	Optional	The type of the filter defined in the filter attribute. Value: `global` or `system` Default value: None
name	Required	The name of the monitor box. Value: String Default value: None
label	Optional	The text to appear on the monitor box. Value: String Default value: None
datasource	Optional	The name of the data source to associate with the monitor box. To specify more than one data source, use a comma-separated list. Value: String Default value: `NCOMS`

Table 13. Attributes of the <monitor> element (continued)

Attribute name	Required or optional	Description
x	Optional	The horizontal position of the monitor box in pixels from the left of the map Value: Integer Default value: None
y	Optional	The vertical position of the monitor box in pixels from the bottom of the map. Value: Integer Default value: None
translucency	Optional	The level of translucency for the monitor box. Value: Integer between 0 and 100 (inclusive). Default value: None
show_shadow	Optional	Specifies whether the monitor box appears on the map with a shadow. Value: `true` or `false` Default value: `false`
font	Optional	The name of the font to use for the monitor box text. Value: String Default value: None
size	Optional	The point size of the font to use for the monitor box text. Value: Integer Default value: None
style	Optional	The text style (for example, bold) for the monitor box text. Value: String Default value: None
color	Optional	The name of the color to use for the button. Value: String Default value: None

Table 13. Attributes of the <monitor> element (continued)

Attribute name	Required or optional	Description
action	Optional	The action that takes place when the user clicks on the monitor box. Value: String. Possible values for opening event lists are as follows: • `"Open URL"` or `"go"` to open a URL. The URL is specified by the url attribute. • `"Active Event List (AEL)"` or `"ael"` to open an Active Event List. • `"Lightweight Event List (LEL)"` to open a Lightweight Event List. • `"Event Table (Table View)"` to open a Table View. • `"Event Viewer"` to open an Event Viewer. • `"Update Event List (using wires)"` to send a NodeClickedOn event using Tivoli Integrated Portal wires. For this option, ensure that a system or custom wire to an Event Viewer or AEL is configured. Default value: None
url	Optional	The uniform resource locator that is the action target for the monitor box. Value: String Default value: None
target	Optional	The target window in the browser for output from the value of the url attribute. When the attribute is not present, output replaces the content of the current browser window. Value: String Default value: `_blank`
flash	Optional	Defines whether the button flashes. Value: `true` or `false` Default value: `false`
w	Optional	The width of the monitor box in pixels. Value: Integer Default value: None
h	Optional	The height of the monitor box in pixels. Value: Integer Default value: None
type	Optional	The type of monitor box, for example a histogram. Value: `lavalamp` or `histogram` Default value: `histogram`

Table 13. Attributes of the <monitor> element (continued)

Attribute name	Required or optional	Description
flash	Optional	Defines whether the monitor box flashes. Value: `true` or `false` Default value: `false`
show_label	Optional	Defines whether to display a label on the monitor box. Value: `true` or `false` Default value: `false`
show_total	Optional	Defines whether to display the total number of alerts on the monitor box. Value: `true` or `false` Default value: `false`
show_highest	Optional	Defines whether to display the highest severity on monitor box. Value: `true` or `false` Default value: `false`
show_lowest	Optional	Defines whether to display the lowest severity on the monitor box. Value: `true` or `false` Default value: `false`
show_metric	Optional	Defines whether to display a metric on the monitor box. Value: `true` or `false` Default value: `false`
show_highest_severity_as _border	Optional	Defines whether to use the color of the highest severity alert that the filter captures as the border of the monitor box. Value: `true` or `false` Default value: `false`
foreground_color	Optional	The name of the color to use for the foreground of the monitor box. Value: String Default value: None
background_color	Optional	The name of the color to use as for the background of the monitor box. Value: String Default value: None

<icon>

The <icon> element is a child of the `<map>` element and defines the characteristics of an icon on a map. The element can occur any number of times and, if present, has the following attributes:

Table 14. Attributes of the <icon> element

Attribute name	Required or optional	Description
filter	Optional	Defines a filter to associate with the icon. Value: String Default value: None
filterType	Optional	The type of the filter defined in the filter attribute. Value: `global` or `system` Default value: None
name	Required	The name of the icon. Value: String Default value: None
label	Optional	The text to appear on the icon. Value: String Default value: None
datasource	Optional	The name of the data source to associate with the icon. To specify more than one data source, use a comma-separated list. Value: String Default value: `NCOMS`
x	Optional	The horizontal position of the icon in pixels from the left of the map Value: Integer Default value: None
y	Optional	The vertical position of the icon in pixels from the bottom of the map. Value: Integer Default value: None
translucency	Optional	The level of translucency for the icon. Value: Integer between 0 and 100 (inclusive). Default value: None
show_shadow	Optional	Specifies whether the icon appears on the map with a shadow. Value: `true` or `false` Default value: `false`
font	Optional	The name of the font to use for the icon text. Value: String Default value: None

Table 14. Attributes of the <icon> element (continued)

Attribute name	Required or optional	Description
font_color	Optional	The name of the color to use for the icon text. Value: String Default value: None
size	Optional	The point size of the font to use for the icon text. Value: Integer Default value: None
style	Optional	The text style (for example, bold) for the icon text. Value: String Default value: None
url	Optional	The uniform resource locator that is the action target for the icon. Value: String Default value: None
action	Optional	The action that takes place when the user clicks on the icon. Value: String. Possible values for opening event lists are as follows: • `"Open URL"` or `"go"` to open a URL. The URL is specified by the url attribute. • `"Active Event List (AEL)"` or `"ael"` to open an Active Event List. • `"Lightweight Event List (LEL)"` to open a Lightweight Event List. • `"Event Table (Table View)"` to open a Table View. • `"Event Viewer"` to open an Event Viewer. • `"Update Event List (using wires)"` to send a NodeClickedOn event using Tivoli Integrated Portal wires. For this option, ensure that a system or custom wire to an Event Viewer or AEL is configured. Default value: None
target	Optional	The target window in the browser for output from the value of the url attribute. When the attribute is not present, output replaces the content of the current browser window. Value: String Default value: `_blank`

Table 14. Attributes of the <icon> element (continued)

Attribute name	Required or optional	Description
flash	Optional	Defines whether the icon flashes. Value: `true` or `false` Default value: `false`
w	Optional	The width of the icon in pixels. Value: Integer Default value: None
h	Optional	The height of the icon in pixels. Value: Integer Default value: None
arc_diameter	Optional	The arc diameter of the icon, in degrees. Value: Integer Default value: None
legend	Optional	The legend for the icon. Value: String Default value: None
image	Optional	The name of an image file to use for the icon. Value: String Default value: None
entity_status_indicator	Optional	Defines the feedback property for the icon. Value: `Highlight Bar`, `Fill Background`, or `Glow Background`. Default value: None

<line>

The `<line>` is a child of the `<map>` element and defines the characteristics of a line on a map. The element can occur any number of times and, if present, has the following attributes:

Table 15. Attributes of the <line> element

Attribute name	Required or optional	Description
filter	Optional	Defines a filter to associate with the line. Value: String Default value: None
filterType	Optional	The type of the filter defined in the filter attribute. Value: `global` or `system` Default value: None

Table 15. Attributes of the <line> element (continued)

Attribute name	Required or optional	Description
name	Required	The name of the line. Value: String Default value: None
label	Optional	The text to appear on the line. Value: String Default value: None
datasource	Optional	The name of the data source to associate with the line. To specify more than one data source, use a comma-separated list. Value: String Default value: `NCOMS`
x	Optional	The horizontal position of the line in pixels from the left of the map Value: Integer Default value: None
y	Optional	The vertical position of the line in pixels from the bottom of the map. Value: Integer Default value: None
translucency	Optional	The level of translucency for the line. Value: Integer between 0 and 100 (inclusive). Default value: None
show_shadow	Optional	Specifies whether the line appears on the map with a shadow. Value: `true` or `false` Default value: `false`

Table 15. Attributes of the <line> element (continued)

Attribute name	Required or optional	Description
action	Optional	The action that takes place when the user clicks on the line. Value: String. Possible values for opening event lists are as follows: • `"Open URL"` or `"go"` to open a URL. The URL is specified by the url attribute. • `"Active Event List (AEL)"` or `"ael"` to open an Active Event List. • `"Lightweight Event List (LEL)"` to open a Lightweight Event List. • `"Event Table (Table View)"` to open a Table View. • `"Event Viewer"` to open an Event Viewer. • `"Update Event List (using wires)"` to send a NodeClickedOn event using Tivoli Integrated Portal wires. For this option, ensure that a system or custom wire to an Event Viewer or AEL is configured. Default value: None
url	Optional	The uniform resource locator that is the action target for the line. Value: String Default value: None
target	Optional	The target window in the browser for output from the value of the url attribute. When the attribute is not present, output replaces the content of the current browser window. Value: String Default value: `_blank`
flash	Optional	Defines whether the line flashes. Value: `true` or `false` Default value: `false`
thickness	Optional	The thickness of the line in pixels. Value: Integer Default value: None
x2	Optional	The ending, horizontal position of the line in pixels from the left of the map. Value: Integer Default value: None

Table 15. Attributes of the <line> element (continued)

Attribute name	Required or optional	Description
y2	Optional	The ending, vertical position of the line in pixels from the bottom of the map. Value: Integer Default value: None
color	Optional	The name of the color to use for the line. Value: String Default value: None

Example

The following example creates a map named `map1` that contains two text labels, three buttons, and three monitor boxes.

```
<methodCall>
 <method methodName="map.createMap">
  <map name="map1">
   <text name="map1" label="map1" x="547" y="30" font="Helvetica" size="22"
    justify="center" style="bi" color="black" rotate="315" />

   <text name="ael_instructions" label="Click  Monitor Box for Tableview"
    x="550" y="141" font="Helvetica" size="10" justify="center"
    style="b" color="black" />

   <button name="button1" label="Active Event List" x="296" y="146"
    "Active Event List (AEL)" filter="Example_LastDay" filtertype="system"
    target="hidden" w="98" h="20" color="lightGray" type="rectangle"
    arc_diameter="20" transparent="false" legend="Label" font="Helvetica"
    size="11" font_color="black" show_shadow="true" />

   <button name="button2" label="Active Event List" x="144" y="146"
    action="Active Event List (AEL)" filter="Example_Critical" filtertype="system"
    target="hidden" w="98" h="20" color="lightGray" type="rectangle"
    arc_diameter="20" transparent="false" legend="Label" font="Helvetica"
    size="11" font_color="black" show_shadow="true" />

   <button name="button3" label="Active Event List" x="21" y="146"
    action="Active Event List (AEL)" filter="Example_Unassigned" filtertype="system"
    target="hidden" w="98" h="20" color="lightGray" type="rectangle"
    arc_diameter="20" transparent="false" legend="Label" font="Helvetica"
    size="11" font_color="black" show_shadow="true" />

   <monitor name="lastday" label="Last Day" x="294" y="10" action="Event Table (Table View)"
    target="_blank" filter="Example_LastDay" filtertype="system" w="100"
    h="126" type="histogram" show_label="true" show_total="true"
    show_highest="false" show_lowest="false" show_metric="true"
    foreground_color="red" background_color="lightGray" font="Helvetica"
    size="10" style="p" show_highest_severity_as_border="false" />

   <monitor name="critical" label="Critical" x="142" y="10" action="Event Table (Table View)"
    target="_blank" filter="Example_Critical" filtertype="system" w="100"
    h="126" type="histogram" show_label="true" show_total="true"
    show_highest="false" show_lowest="false" show_metric="true"
    foreground_color="black" background_color="lightGray" font="Helvetica"
    size="10" style="p" show_highest_severity_as_border="false" />

   <monitor name="unassigned" label="Unassigned" x="17" y="10" action="Event Table (Table View)"
    target="_blank" filter="Example_Unassigned" filtertype="system" w="100"
```

```
    h="126" type="histogram" show_label="true" show_total="true"
    show_highest="false" show_lowest="false" show_metric="true"
    foreground_color="black" background_color="lightGray" font="Helvetica"
    size="10" style="p" show_highest_severity_as_border="false" />
  </map>
 </method>
</methodCall>
```

Create or replace a map

The format of the <method> element when creating or replacing a map is:

```
<method methodName="map.createOrReplaceMap">
```

Use this method to replace an existing map or create one if it does not already exist. The <method> element contains one or more <map> elements each of which define the characteristics of a map. Each <map> element contains any number of the <text>, <button>, <monitor>, <icon>, and <line> elements. For more information, see "<map>" on page 27.

This example creates or replaces a map named map3 that has two monitor boxes and a text label.

```
<methodCall>
    <method methodName="map.createOrReplaceMap">
     <map name="map3">
      <text name="ael_instructions" label="Click  Monitor Box for Tableview"
       x="550" y="141" font="Helvetica" size="10" justify="center"
       style="b" color="black" />

      <monitor name="critical" label="Critical" x="142" y="10" action="table"
       target="_blank" filter="Example_Critical" filtertype="system" w="100"
       h="126" type="histogram" show_label="true" show_total="true"
       show_highest="false" show_lowest="false" show_metric="true"
       foreground_color="black" background_color="lightGray" font="Helvetica"
       size="10" style="p" show_highest_severity_as_border="false" />

      <monitor name="unassigned" label="Unassigned" x="17" y="10" action="table"
       target="_blank" filter="Example_Unassigned" filtertype="system" w="100"
       h="126" type="histogram" show_label="true" show_total="true"
       show_highest="false" show_lowest="false" show_metric="true"
       foreground_color="black" background_color="lightGray" font="Helvetica"
       size="10" style="p" show_highest_severity_as_border="false" />
     </map>
    </method>
</methodCall>
```

Delete a map

The format of the <method> element when deleting a map is:

```
<method methodName="map.deleteMap">
```

Use this method to delete an existing map. The <method> element contains one or more <map> elements each of which identifies a map to delete. In the <map> element include only the name attribute. For more information, see "<map>" on page 27.

This example deletes the map named map2.

```
<methodCall>
    <method methodName="map.deleteMap">
        <map name="map2">
        </map>
      </method>
</methodCall>
```

Get a list of maps

The format of the <method> element for getting a list of maps is:

```
<method methodName="map.getList">
```

The method returns a list of all the maps defined in the Web GUI.

```
<methodCall>
    <method methodName="map.getList">
    </method>
</methodCall>
```

Modify a map

The format of the <method> element when modifying a map is:

```
<method methodName="map.modifyMap">
```

Use this method to modify an existing map. The <method> element contains one or more <map> elements each of which identifies the map and the characteristics to change. Each <map> element contains any number of the <text>, <button>, <monitor>, <icon>, and <line> elements. For more information, see "<map>" on page 27.

This example makes the following changes to a map named map1:

- Set the background color to white.
- Change the width of the map to 850 pixels
- Change the height of the map to 490 pixels.

```
<methodCall>
    <method methodName="map.modifyMap">
      <map name="map1" bgImage="" bgColor="white" w="850" h="490" >
      </map>
    </method>
</methodCall>
```

Add a map visual

The format of the <method> element when adding a visual to a map is:

```
<method methodName="map.addMapVisual">
```

Use this method to add one or more objects to an existing map. The <method> element contains one or more <map> elements that identify maps to add new objects to. Each <map> element contains any number of <text>, <button>, <monitor>, <icon>, and <line> elements each of which define a new object to add to the map. For more information, see "<map>" on page 27.

This example adds a text label and a monitor box to the map named map2.

```
<methodCall>
 <method methodName="map.addMapVisual">
  <map name="map2">
   <text name="map2" label="map2" x="147" y="30" font="Helvetica" size="22"
    justify="center" style="bi" color="black" />

   <monitor name="lastday" label="Last Day" x="131" y="92" action="table"
    url="" target="_blank" filter="ExampleLastDay" filtertype="system"
    w="100" h="126" type="histogram" show_label="true" show_total="true"
    show_highest="false" show_lowest="false" show_metric="true"
    foreground_color="red" background_color="lightGray" font="Helvetica"
```

```
    size="10" style="p" show_highest_severity_as_border="false" />
  </map>
 </method>
</methodCall>
```

Add or replace a map visual

The format of the <method> element when adding or replacing a map object is:

```
<method methodName="map.createOrReplaceMapVisual">
```

Use this method to add a new object to a map, or replace the object if it already exists. The <method> element contains one or more <map> elements whose contents are to be modified. Each <map> element contains any number of <text>, <button>, <monitor>, <icon>, and <line> elements each of which defines an object to add or replace. in the <map> element include only the name attribute. For more information, see "<map>" on page 27.

This example creates or replaces an icon on the map named map2.

```
<methodCall>
 <method methodName="map.createOrReplaceMapVisual">
  <map name="map2">
   <icon name="visual_3" label="Active Icon" x="75" y="233" action="ael"
    filter="Example_Critical" filtertype="system" target="hidden" w="30" h="19"
    type="rectangle" arc_diameter="10" legend="None" font="Helvetica"
    size="10" font_color="black" image="blocks.png"
    entity_status_indicator="Glow Background"/>
  </map>
 </method>
</methodCall>
```

Delete a map visual

The format of the <method> element when deleting an object from a map is:

```
<method methodName="map.deleteMapVisual">
```

Use this method to remove one or more objects from a map. The <method> element contains one or more <map> elements each of which identify a map to modify. Each <map> element can contains one or more <text>, <button>, <monitor>, <icon>, or <line> elements that identify the object to remove. In all of these elements, include only the name attribute. For more information, see "<map>" on page 27.

This example removes a text label name visual_4 from a map named map1.

```
<methodCall>
 <method methodName="map.deleteMapVisual">
  <map name="map1">
   <text name="visual_4"  />
  </map>
 </method>
</methodCall>
```

Modify a map visual

The format of the <method> element when modifying an object on a map is:

```
<method methodName="map.modifyMapVisual">
```

Use this method to change the characteristics of an object on a map. The <method> element contains one or more <map> elements that identify the maps to modify. Each <map> element contains one or more <text>, <button>, <monitor>, <icon>, and <line> elements which define the objects to modify. In the <map> element include only the name attribute. For more information, see "<map>" on page 27.

This example makes the following changes to a text label named visual_1 on a map named map1:

- Set the label to My Active Button
- Change the horizontal position of the button to 230 pixels
- Change the vertical position of the button to 188 pixels

```
<methodCall>
 <method methodName="map.modifyMapVisual">
  <map name="map1">
   <button name="visual_1" label="My Active Button" x="230" y="188"  />
  </map>
 </method>
</methodCall>
```

Resource requests

Resources are images that can appear on Web GUI maps. There are functions to add and remove resources as well as obtain a list of the resources defined on the Web GUI server. WAAPI provides four methods for working with map resources: Adding resources, creating or replacing resources, removing resources, and getting a list of resources.

Add a resource

The format of the <method> element for adding a resource is:

```
<method methodName="resource.addResource">
```

Use this method to add a resource to the system. The <method> element contains one or more <resources> elements each of which defines a map to add the resource to. The <resources> element contains one or more <resource> elements each of which identifies a resource.

<resources>

The <resources> element defines a set of resources for a map and has the following attribute:

Table 16. Attributes of the <resources> element

Attribute name	Required or optional	Description
mapName	Required	The name of a map to add the resource to. Value: String Default value: None

<resource>

The <resource> element is a child element of <resources>. The element defines a resource for a map and has the following attribute:

Table 17. Attributes of the <resource> element

Attribute name	Required or optional	Description
name	Required	The name of the resource to add to the map. Value: String Default value: None

Example

This example adds an image named ny.gif to a map named map1.

```
<methodCall>
 <method methodName="resource.addResource">
  <resources mapName="map1">
   <resource name="ny.gif" />
  </resources>
 </method>
</methodCall>
```

Create or replace a resource

The format of the <method> element for creating or replacing a resource is:

```
<method methodName="createOrReplaceResource">
```

Use this method to add a resource to a map, if the map does not contain the resource, or to replace the existing instance of the resource on the map. The <method> element contains any number of <resources> elements that identify the maps to modify. For more information, see "<resources>" on page 46. Each <resources> element contains one or more <resource> elements that each identify the resources to add or replace. For more information, see "<resource>."

Example

This example adds or replaces an image named ny.gif to a map named map1.

```
<methodCall>
 <method methodName="resource.creatOrReplaceResource">
  <resources mapName="map1">
   <resource name="ny.gif" />
  </resources>
 </method>
</methodCall>
```

Remove a resource

The format of the <method> element for removing a resource is:

```
<method methodName="removeResource">
```

Use this method to remove a resource from a map. The <method> element contains any number of <resources> elements that identify the maps to modify. For more information, see "<resources>" on page 46. Each <resources> element contains one or more <resource> elements that each identify the resources to add or replace. For more information, see "<resource>" on page 47.

Example

This example removes the resource ny.gif from a map named map1.

```
<methodCall>
 <method methodName="resource.removeResource">
  <resources mapName="map1">
   <resource name="ny.gif" />
  </resources>
 </method>
</methodCall>
```

Get a list of resources

The format of the <method> element for getting a list of resources is:

```
<method methodName="resource.getList">
```

This method returns a list of all the resources defined on the Web GUI server and has no child elements. Each map has its own section in the listing.

Example

```
<methodCall>
 <method methodName="resource.getList">
 </method>
</methodCall>
```

File requests

File requests enable you to work on files and directories on the Web GUI server. You can create and delete files and directories. WAAPI provides six methods for working with files: Adding directories, adding files, creating or replacing files, deleting files, removing directories, and recursively removing directories.

Add a directory

The format of the <method> element for adding a directory to the Web GUI server is:

```
<method methodName="file.addDir">
```

Use this method to create a new directory on the Web GUI server. The <method> element contains one or more <file> elements each of which defines a directory to create on the system. In each <file> element include the dirName attribute to define the directory to create.

- "<file>" on page 49
- "Example" on page 49

<file>

The <file> element defines the characteristics of a file or directory to work with. The element has the following attributes:

Table 18. Attributes of the <file> element

Attribute name	Required or optional	Description
fileName	Optional	The name of a file to work with. Value: String Default value: None
dirName	Optional	The specification of a directory to work with. Value: String Default value: None
toDir	Optional	The destination directory for a command. Value: String Default value: None
fromDir	Optional	The source directory for a command. Value: String Default value: None

Example

This example creates a directory named `data` on the `C:` drive of a Windows system.

```
<methodCall>
 <method methodName="file.addDir">
  <file dirName="C:\data">
  </file>
 </method>
</methodCall
```

Add a file

The format of the `<method>` element for adding a file to the Web GUI server is:

```
<method methodName="file.addFile">
```

Use this method to create a file on the Web GUI server. The `<method>` element contains one or more `<file>` elements that each define the name and location of a file to create. The content of the `<file>` element becomes the content of the new file. For more information, see "<file>."

Example

This example creates a file named `data.txt` in the directory `C:\data\` on a Windows system.

```
<methodCall>
 <method methodName="file.addFile">
  <file fileName="data.txt" toDir="C:\data">
  </file>
 </method>
</methodCall>
```

Create or replace a file

The format of the <method> element for creating or replacing a file is:

```
<method methodName="file.createOrReplaceFile">
```

Use this method to replace an existing file with the same name or to create the file, if it does not exist. The <method> element contains one or more <file> elements each of which identifies a file to create or replace. The content of the <file> element becomes the content of the new file. For more information, see “<file>” on page 49.

Example

This example creates or replaces a file named hello.txt in the directory C:\data and sets the contents of the file to the text "Hello world".

```
<methodCall>
 <method methodName="file.createOrReplaceFile">
  <file fileName="hello.txt" toDir="C:\data">
  </file>
 </method>
<methodCall>
```

Delete a file

The format of the <method> element for deleting a file is:

```
<method methodName="file.deleteFile">
```

Use this method to delete a file from the Web GUI server. The <method> element contains one or more <file> elements each of which defines the name and location of a file to delete. For more information, see “<file>” on page 49.

Example

This example deletes the file hello.txt from the directory C:\data.

```
<methodCall>
 <method methodName="file.deleteFile">
  <file fileName="hello.txt" fromDir="C:\data">
  </file>
 </method>
</methodCall>
```

Remove a directory

The format of the <method> element for removing a directory is:

```
<method methodName="deleteDir">
```

Use this method to remove a directory from the Web GUI server. The <method> element contains one or more <file> elements each of which defines a directory to remove. Each directory must be empty. For more information, see “<file>” on page 49.

Example

This example removes the directory named C:\test.

```
<methodCall>
 <method methodName="deleteDir">
  <file dirName="test">
  </file>
 </method>
</methodCall>
```

Recursively remove a directory

The format of the <method> element for removing a directory recursively is:

```
<method methodName="file.recurseRemove">
```

Use this method to remove a directory and all its contents. This method is useful for deleting directory trees. The <method> element contains one or more <file> elements each of which defines a directory to delete, along with all its contents. For more information, see "<file>" on page 49.

Example

This example recursively removes the directory named C:\data.

```
<methodCall>
 <method methodName="recurseRemove">
  <file dirName="C:\data">
  </file>
 </method>
</methodCall>
```

Menu requests

Menus provide users with access to Web GUI functions. There are functions to create, modify, and delete menus. You can also get a list of all the menus on the system. WAAPI provides five methods for working with menus: Creating a menu, creating or replacing a menu, modifying a menu, deleting a menu, and getting a list of menus.

Create a menu

The format of the <method> element for creating a menu is:

```
<method methodName="menu.createMenu">
```

Use this method to create a new menu that can contain tools, submenus, and separators. The <method> element contains one or more <supermenu> elements. Each <supermenu> element contains any number of <separator>, <menu>, and <tool> elements that define the content of the menu.

- "<supermenu>"
  - "<separator>" on page 52
  - "<menu>" on page 52
  - "<tool>" on page 52
- "Example" on page 53

<supermenu>

The <supermenu> element is the container for a menu and has the following attributes:

Table 19. Attributes of the <supermenu> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a menu. Value: String Default value: None
label	Optional	The title of the menu as it appears in the AEL. Value: String Default value: The value of the name attribute.
mnemonic	Optional	A single character that allows the user to select the menu using the Alt key. Value: A single alphabetic character Default value: None

<separator>

The `<separator>` element is a child of the `<supermenu>` element and defines the location of a separator line in a menu and can occur any number of times. The element has no content and no attributes.

<menu>

The `<menu>` element is a child of the `<supermenu>` element and defines the location of a submenu. The element can occur any number of times and has the following attributes:

Table 20. Attributes of the <menu> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a menu. Value: String Default value: None
label	Required	The title of the menu as it appears in the AEL. Value: String Default value: None.
mnemonic	Optional	A single character that allows the user to select the menu using the Alt key. Value: A single alphabetic character Default value: None

<tool>

The `<tool>` element is a child of the `<supermenu>` element that defines a tool to appear as an option on the menu. The element can appear any number of times and has the following attributes:

Table 21. Attributes of the <tool> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a tool. Value: String Default value: None
label	Required	The title of the tool as it appears on the menu in the AEL. Value: String Default value: None.
mnemonic	Optional	A single character that allows the user to select the menu that holds the tool using the Alt key. Value: A single alphabetic character Default value: None
shortcut	Optional	A single character that allows the user to select the tool using the Ctrl key. Value: A single alphabetic character Default value: None

Example

This example creates empty menus named `mentu1` and `toolMenu1` that other functions add items to.

```
<methodCall>
 <method methodName="menu.createMenu">
  <supermenu name="menu1" >
  </supermenu>
  <supermenu name="toolMenu1">
  </supermenu>
 </method>
</methodCall>
```

Create or replace a menu

The format of the `<method>` element for creating or replacing a menu is:

```
<method methodName="menu.createOrReplaceMenu">
```

Use this method to create a new menu or replace an existing one that can contain tools, submenus, and separators. The `<method>` element contains one or more `<supermenu>` elements. Each `<supermenu>` element contains any number of `<separator>`, `<menu>`, and `<tool>` elements that define the content of the menu. For more information, see “<supermenu>” on page 51.

Example

This example creates or replaces a menu named `toolMenu2` and adds two tools to it.

```
<methodCall>
 <method methodName="menu.createOrReplaceMenu">
  <supermenu name="toolMenu2">
   <tool shortcut="" label="Ping" name="Ping" mnemonic="" />
```

```
   <tool shortcut="" label="commandTool1" name="commandTool1" mnemonic="" />
  </supermenu>
 <method>
</methodCall>
```

Delete a menu

The format of the <method> element for deleting a menu is:

```
<method methodName="menu.deleteMenu">
```

Use this method to delete an existing menu. The <method> element contains one or more <supermenu> elements each of which identifies a menu to delete. In the <supermenu> element include only the name attribute.

Example

This example deletes a menu named menu1

```
<methodCall>
 <method methodName="menu.deleteMenu">
  <supermenu name="menu1">
  </supermenu>
 </method>
</methodCall>
```

Get a list of menus

The format of the <method> element for getting a list of menus is:

```
<method methodName="menu.getList">
```

The method returns a list of all the menus defined in the Web GUI.

Example

```
<methodCall>
 <method methodName="menu.getList">
 </method>
</methodCall>
```

Modify a menu

The format of the <method> element for modifying a menu is:

```
<method methodName="menu.modifyMenu">
```

Use this method to modify an existing menu. The <method> element contains one or more <supermenu> elements. Each <supermenu> element contains any number of <separator>, <menu>, and <tool> elements that define the content of the menu.

Example

This example adds a menu and two tools to menu1 and then adds that menu, together with further tools, to toolMenu2.

```
<methodCall>
 <method methodName="menu.modifyMenu">

  <supermenu name="menu1" mnemonic="n" >
   <menu name="alerts" label="Alerts" mnemonic="a" />
   <separator />
   <tool name="acknowledge" label="Acknowledged" mnemonic="a"
    shortcut="ctrl+a" />
   <separator />
   <tool name="prioritise" label="Prioritize" mnemonic="p" shortcut="" />
```

```
  </supermenu>

  <supermenu name="toolMenu1">
   <tool shortcut="" label="Ping" name="Ping" mnemonic="" />
   <separator />
   <menu name="menu1" label="menu1"/>
   <separator />
   <tool shortcut="" label="commandTool1" name="commandTool1" mnemonic="" />
  </supermenu>
 </method>
</methodCall>
```

Tool requests

Tools provide enhanced abilities for user to manage events in the AEL. There are functions to create, modify, and delete menus. You can also get a list of all the tools defined on the system. WAAPI provides five methods for working with tools: Creating a tool, creating or replacing a tool, modifying a tool, deleting a tool, and getting a list of tools.

Create a tool

The format of the `<method>` element for creating a tool is:

```
<method methodName="createTool">
```

Each of those elements defines a particular tool.

- "<tool:tool>"
  - "<tool:access>" on page 56
    - "<tool:osfield>" on page 56
      - "<tool:criterion>" on page 56
        - "<tool:equals>" on page 56
      - "<tool:security>" on page 57

    "<tool:sql>" on page 57
  - "<tool:journal>" on page 57
  - "<tool:cgiurl>" on page 58
    - "<tool:fieldlist>" on page 58
      - "<tool:field>" on page 58
  - "<tool:cmdline>" on page 59
    - "<tool:command>" on page 59
  - "<tool:script>" on page 59
- "Usage notes" on page 60
- "Example: SQL tool" on page 60
- "Example: CGI/URL tool" on page 61
- "Script tool" on page 61

<tool:tool>

The `<tool:tool>` element is the container for a tool and can occur any number of times. The element has the following attributes:

Table 22. Attributes of the <tool:tool> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a tool. Value: String Default value: None
datasource	Optional	The data sources that a tool uses. To specify more than one data source use a comma-separated list. Value: String Default value: The default data source

<tool:access>

The `<tool:access>` element is a child of the `<tool:tool>` element and defines the access criteria for a tool. The element has no attributes but contains the following child elements in this order:

- `<tool:osfield>`
- `<tool:security>`

<tool:osfield>

The `<tool:osfield>` contains the ObjectServer fields that a tool uses. The element has no attributes but contains any number of `<tool:criterion>` elements.

<tool:criterion>

The `<tool:criterion>` element is a child of the `<tool:osfield>` and `<tool:security>` elements. It defines an access criterion for a tool and contains one or more `<tool:equals>` elements. In addition, the element has the following attribute:

Table 23. Attributes of the <tool:criterion> element

Attribute name	Required or optional	Description
name	Required	The type of access for a tool. Value: `Class` or `Group`. When this element is a child of `<tool:osfield>` the value of this attribute must be `Class`. When the element is a child of `<tool:security>` the value of the attribute must be `Group`. Default value: None

<tool:equals>

The `<tool:equals>` element is a child of `<tool:criterion>` and assigns a value to an object defined by its parent element. The element has no content but has the following attributes:

Table 24. Attributes of the <tool:equals> element

Attribute name	Required or optional	Description
value	Required	Provides a value for the access criterion. Value: String Default value: None
datasource	Optional	The data sources that the criterion uses. To specify more than one data source use a comma-separated list. Value: String Default value: The default data source

<tool:security>

The `<tool:security>` element defines the access criteria for an AEL tool. The tool contains any number of `<tool:criterion>` elements that define the access criteria. The element has no attributes. For more information about the `<tool:criterion>` element, see "<tool:criterion>" on page 56.

<tool:sql>

The `<tool:sql>` element contains the SQL command and associated options for an SQL AEL tool. The element is empty but has the following attributes:

Table 25. Attributes of the <tool:sql> element

Attribute name	Required or optional	Description
foreach	Optional	Defines whether a tool acts on each of the selected rows in the AEL. Value: `true` or `false` Default value: `false`
command	Required	The SQL command for the tool. Value: String Default value: None

<tool:journal>

The `<tool:journal>` element contains the journal entry made when an AEL tool is run. The element is empty but has the following attributes:

Table 26. Attributes of the <tool:journal> element

Attribute name	Required or optional	Description
foreach	Optional	Defines whether a tool acts on each of the selected rows in the AEL. Value: `true` or `false` Default value: `false`

Table 26. Attributes of the <tool:journal> element (continued)

Attribute name	Required or optional	Description
entry	Required	The text for the journal entry. Value: String Default value: None

<tool:cgiurl>

The `<tool:cgiurl>` element defines a CGI/URL tool. The element contains the `<tool:fieldlist>` element and has the following attributes:

Table 27. Attributes of the <tool:cgiurl> element

Attribute name	Required or optional	Description
foreach	Optional	Defines whether a tool acts on each of the selected rows in the AEL. Value: `true` or `false` Default value: `false`
windowforeach	Optional	Defines whether to display an output window for each row when the tool is run against the AEL. Value: `true` or `false` Default value: `false`
method	Optional	The method that the tool uses to send data to the server. Value: `GET` or `POST` Default value: `GET`
target	Optional	The target window in the browser for output from tool linked to a URL, specified in the url attribute. Value: String Default value: `_blank`
url	Required	The uniform resource locator that is the action target for the tool. Value: String Default value: None

<tool:fieldlist>

The `<tool:fieldlist>` element is a child of `<tool:cgiurl>` and defines the ObjectServer fields to use with the tool. The element contains one or more `<tool:field>` elements each identifying an ObjectServer field and has no attributes.

<tool:field>

The `<tool:field>` element is a child of `<tool:fieldlist>` and defines the name of an ObjectServer field to use in the tool. the element is empty but has the following attribute:

Table 28. Attributes of the <tool:field> element

Attribute name	Required or optional	Description
name	Required	The name of an ObjectServer field. Value: String Default value: None

<tool:cmdline>

The `<tool:cmdline>` element defines a command line tool. The element contains one or more `<tool:command>` elements and has no attributes.

<tool:command>

The `<tool:command>` element is a child of `<tool:cmdline>` and defines the platform-specific command associated with a command line tool. The element is empty but has the following attributes:

Table 29. Attributes of the <tool:command> element

Attribute name	Required or optional	Description
enabled	Optional	Defines whether is enabled or disabled. It operates in conjunction with the platform attribute to determine if a tool is available on a specific platform Value: `true` or `false` Default value: `false`
platform	Required	Identifies the platform that the tool applies to. It operates in conjunction with the enabled attribute to determine if a tool is available on a specific platform. Value: `Windows`, `Solaris`, `Linux`, `HPUX`, or `AIX` Default value: None
executable	Required	The command that executes when the tool is run. Value: String Default value: None
foreach	Optional	Defines whether a tool acts on each of the selected rows in the AEL. Value: `true` or `false` Default value: `false`

<tool:script>

The `<tool:script>` element defines a script tool. The element is empty but has the following attributes:

Table 30. Attributes of the <tool:script> element

Attribute name	Required or optional	Description
foreach	Optional	Defines whether a tool acts on each of the selected rows in the AEL. Value: `true` or `false` Default value: `false`
command	Required	The script for the tool. Value: String Default value: None

Usage notes

The `<tool:journal>` element is typically used in conjunction with other tools to create journal entries recording what the tools have done. For example, you can use the `<tool:journal>` element to record the action taken by a `<tool:sql>` element.

To create a command line tool that can run on any platform in your organization, use multiple `<tool:cmdline>` elements. Each of these contains a platform-specific command all of which achieve the same, or similar, result. For example, you could have and element for Windows and another Linux. The example in "Create or replace a tool" on page 62 shows this technique.

Example: SQL tool

The following example creates a SQL tool named `sqlSample1`.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">
 <method methodName="tool.createTool">
  <!--
   SQL tool with Class access criteria set to Tivoli TEC (Oracle) (7450)
   and Tivoli TEC (Sybase) (7500), and Group access criteria set to restricted
   and Desktop.
  -->
  <tool:tool name="sqlSample1">
   <tool:access>
    <tool:osfield>
     <tool:criterion name="Class">
      <tool:equals value="7450"/>
      <tool:equals value="7500"/>
     </tool:criterion>
    </tool:osfield>
    <tool:security>
     <tool:criterion name="Group">
      <tool:equals value="restricted"/>
      <tool:equals value="Desktop"/>
     </tool:criterion>
    </tool:security>
   </tool:access>
   <tool:sql foreach="true"
    command="update alerts.status set Summary='sqlTest1' where Serial=@Serial;"
   />
   <tool:journal foreach="true"
    entry="Tool executed on event Class CONVERSION(@Class)."/>
  </tool:tool>
 </method>
</methodCall>
```

Example: CGI/URL tool

The following example creates a CGI/URL tool named cmsSample1.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">
 <method methodName="tool.createTool">
  <!--
   Command Line tool with Group access criteria set to restricted and Desktop.
   Class access criteria is undefined, therefore executable against events of
   any Class.
  -->
  <tool:tool name="cmdSample">
   <tool:access>
    <tool:osfield/>
    <tool:security>
     <tool:criterion name="Group">
      <tool:equals value="restricted"/>
      <tool:equals value="Desktop"/>
     </tool:criterion>
    </tool:security>
   </tool:access>
   <tool:cmdline>
    <tool:command foreach="true" enabled="true" platform="Windows"
     executable="start cmd /k c:\temp\sample.bat {@Class} {@Node}"/>
    <tool:command foreach="true" enabled="true" platform="Solaris"
     executable="xterm -e /bin/sh -c '/tmp/sample.sh {@Class} {@Node}; read a'"
    />
    <tool:command foreach="true" enabled="false" platform="AIX"
     executable="xterm -e /bin/sh -c '/tmp/sample.sh {@Class} {@Node}; read a'"
    />
   </tool:cmdline>
  </tool:tool>
 </method>
</methodCall>
```

Script tool

The following example creates a script tool named scriptSample1.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">
 <method methodName="tool.createTool">
  <!--
   Script tool with Group access criteria set to restricted and Desktop. Class
   access criteria is undefined, therefore executable against events of
   any Class.
  -->
  <tool:tool name="scriptSample">
   <tool:access>
    <tool:osfield/>
    <tool:security>
     <tool:criterion name="Group">
      <tool:equals value="restricted"/>
      <tool:equals value="Desktop"/>
     </tool:criterion>
    </tool:security>
   </tool:access>
   <tool:script foreach="false"  command="alert('{@Node}');"/>
  </tool:tool>
 </method>
</methodCall>
```

Create or replace a tool

The format of the <method> element for creating or replacing a tool is:

```
<method methodName="createOrReplaceTool">
```

Use this method to create a new tool or replace an existing tool. The <method> element contains one or more <tool:tool> elements. Each <tool:tool> element contains a <tool:access> element followed by any number of one of the following elements:

- <tool:sql>
- <tool:journal>
- <tool:cgiurl>
- <tool:cmdline>
- <tool:script>

Each of those elements defines a particular tool. For more information, see "<tool:tool>" on page 55.

Example

This example creates or replaces a tool named cmdSample.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">
 <method methodName="tool.createOrReplaceTool">
  <!-- Create or replace a tool called cmdSample. -->
  <tool:tool name="cmdSample">
   <tool:access>
    <tool:osfield/>
    <tool:security/>
   </tool:access>
   <tool:cmdline>
    <tool:command foreach="true" enabled="true" platform="Windows"
     executable="start cmd /k c:\temp\sample.bat {@Class} {@Node}"/>
    <tool:command foreach="true" enabled="true" platform="Linux"
     executable="xterm -e /bin/sh -c '/tmp/sample.sh {@Class} {@Node}; read a'"/>
    <tool:command foreach="true" enabled="true" platform="AIX"
     executable="xterm -e /bin/sh -c '/tmp/sample.sh {@Class} {@Node}; read a'"/>
   </tool:cmdline>
  </tool:tool>
 </method>
</methodCall>
```

Delete a tool

The format of the <method> element when deleting a tool is:

```
<method methodName="tool.deleteTool">
```

Use this method to delete an existing tool. The <method> element contains one or more <tool:tool> elements each of which identifies a tool to delete. In the <tool:tool> element include only the name attribute.

Example

The following example deletes the tools named sqlSample1, sqlSample2, cgiSample, cmdSample, scriptSample, and scriptSample2.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">

 <!-- Delete tools if they exist. -->
 <method methodName="tool.deleteTool">
   <tool:tool name="sqlSample1"/>
```

```
   <tool:tool name="sqlSample2"/>
   <tool:tool name="cgiSample"/>
   <tool:tool name="cmdSample"/>
   <tool:tool name="scriptSample"/>
   <tool:tool name="scriptSample2"/>
 </method>
</methodCall>
```

Get a list of tools

The format of the <method> element for getting a list of tools is:

```
<method methodName="tool.getList">
```

The method returns a list of all the tools defined in the Web GUI.

Example

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">

 <!-- Get a list of tools available in the server. -->
 <method methodName="tool.getList"/>
 </method>
</methodCall>
```

Modify a tool

The format of the <method> element for modifying a tool is:

```
<method methodName="modifyTool">
```

Use this method to modify an existing tool. The <method> element contains one or more <tool:tool> elements. Each <tool:tool> element contains a <tool:access> element followed by any number of one of the following elements:

- <tool:sql>
- <tool:journal>
- <tool:cgiurl>
- <tool:cmdline>
- <tool:script>

For more information, see "<tool:tool>" on page 55.

Example

The following example modifies tools named sqlSample1 and cgiSample.

```
<methodCall xmlns:tool="http://www.ibm.com/tivoli/netcool/webtop/tools/2.1">
 <method methodName="tool.modifyTool">
  <tool:tool name="sqlSample1">
   <tool:access>
    <tool:osfield>
     <tool:criterion name="Class">
      <tool:equals value="7450"/>
     </tool:criterion>
    </tool:osfield>
    <tool:security/>
   </tool:access>
   <tool:sql foreach="true"
    command="update alerts.status set Summary='sqlTest1' where Serial=@Serial;"
   />
   <tool:journal foreach="true" entry="Tool executed on event Class @Class."/>
  </tool:tool>

  <tool:tool name="cgiSample">
```

```
   <tool:access>
    <tool:osfield>
     <tool:criterion name="Class">
      <tool:equals value="7500"/>
     </tool:criterion>
    </tool:osfield>
    <tool:security/>
   </tool:access>
   <tool:cgiurl foreach="true" windowforeach="true" target="_blank"
    method="GET" url="$(SERVER)/cgi-bin/sample.cgi">
    <tool:fieldlist>
     <tool:field name="Class"/>
     <tool:field name="Node"/>
    </tool:fieldlist>
   </tool:cgiurl>
  </tool:tool>
 </method>
</methodCall>
```

Prompt requests

Prompts are associated with certain types of tools. They generate a prompt window or a pop-up menu for users to enter or select information that a tool requires. There are functions to create. modify, and delete prompts. In addition you can obtain a list of prompts defined on the server. WAAPI provides five methods for working with prompts: Creating a prompt, creating or replacing a prompt, modifying a prompt, deleting a prompt, and getting a list of prompts.

The WAAPI prompts file contains several example prompts and is located in *webgui-home*/waapi/etc/samples/samplerequest_prompt.xml.

Create or replace a prompt

The format of the <method> element for creating or replacing a prompt is:

```
<method methodName="prompt.creatOrReplacePrompt">
```

Use this method to create a new prompt or replace an existing one. The <method> elements contains one or more <prompt:prompt> elements each of which defines the characteristics of a new prompt. The <prompt:prompt> element contains an optional <prompt:parameters> element followed by an optional <prompt:choice> element. For more information, see <prompt:prompt>, and for more information about how to use the <prompt:prompt> element and its subelements, see Usage notes.

Example

This example replaces the definition of the existing prompt named testFormattedString.

```
<methodCall xmlns:prompt="http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2">
  <method methodName="prompt.createOrReplacePrompt">
    <!-- testFormattedString exists, so it is replaced with new settings. -->
    <prompt:prompt type="FormattedString" name="testFormattedString">
      <prompt:parameters label="Type in the ticket ID (3 letters, 3 digits)"
       order="180" errorMessage="Invalid ticket ID format" localized="false"
       defaultValue="Bbc123">
       <prompt:additionalParams>
        <prompt:param name="format" value="^[a-zA-Z]{3}\d{3}$"/>
       </prompt:additionalParams>
      </prompt:parameters>
    </prompt:prompt>
  </method>
</methodCall>
```

Delete a prompt

The format of the <method> element for deleting a prompt is:

```
<method methodName="prompt.deletePrompt">
```

Use this method to delete an existing prompt. The <method> element contains one or more <prompt:prompt> elements each of which identifies a prompt to delete. In the <prompt:prompt> element include only the name attribute.

Example

This example deletes te prompt named testString.

```
<methodCall xmlns:prompt="http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2">
  <method methodName="prompt.deletePrompt">
     <prompt:prompt name="testString"/>
   </method>
</methodCall>
```

Get a list of prompts

The format of the <method> element for getting a list of prompts is:

```
<method methodName="prompt.getList" />
```

Use this method to obtain a list containing the names of the defined prompts.

Example

```
<methodCall xmlns:prompt="http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2">

    <!-- Get a list of prompts available in the server. -->
    <method methodName="prompt.getList"/>
</methodCall>
```

Modify a prompt

The format of the <method> element for modifying a prompt is:

```
<method methodName="prompt.modifyPrompt">
```

Use this method to modify an existing prompt. The <method> elements contains one or more <prompt:prompt> elements each of which defines the characteristics of a new prompt. The <prompt:prompt> element contains an optional <prompt:parameters> element followed by an optional <prompt:choice> element. For more information, see <prompt:prompt>, and for more information about how to use the <prompt:prompt> element and its subelements, see Usage notes.

Example

This example modifies the prompt named testFixedChoice.

```
<methodCall xmlns:prompt="http://www.ibm.com/tivoli/netcool/webtop/prompts/2.2">
  <method methodName="prompt.modifyPrompt">
    <prompt:prompt type="FixedChoice" name="testFixedChoice">
      <prompt:parameters label="Select a fixed option" order="160"
       localized="false" defaultValue="fixed6"/>
      <prompt:choice>
        <prompt:item value="fixed4" label="Fixed Four"/>
        <prompt:item value="fixed5" label="Fixed Five"/>
        <prompt:item value="fixed6" label="Fixed Six"/>
      </prompt:choice>
    </prompt:prompt>
 </method>
</methodCall>
```

CGI requests

CGI requests operate on CGI scripts used as tools in the Web GUI. There are functions to register, modify, and unregister a script. WAAPI provides four methods for working with CGI scripts: Registering a CGI script, creating or replacing a CGI script, modifying a CGI script, and unregistering a CGI script.

Register a CGI script

The format of the <method> element for registering a CGI script is:

```
<method methodName="cgi.registerCGI">
```

Use this method to register a CGI script with the Web GUI server. The <method> element contains one or more <cgi> elements each defining a script to register.

- "<cgi>"
- "Example"

<cgi>

The <cgi> element defines the characteristics of a CGI script and has the following attributes:

Table 31. Attributes of the <cgi> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a CGI script. Value: String Default value: None
useSmartPageCommands	Optional	Defines whether to use SmartPage commands when running the script. Value: `true` or `false` Default value: `false`
fileName	Optional	The path name of the file that contains the script. Value: String Default value: If this attribute is omitted, the content of the <cgi> element contains the script.

Example

This example registers a script named `myping` contained in a file named `nco_myping.cgi`.

```
<methodCall>
 <method methodName="cgi.registerCGI">
  <cgi name="myping" acl="*" useSmartPageCommands="true"
   fileName="/home/nco_myping.cgi">
  </cgi>
 </method>
</methodCall>
```

Create or replace a CGI script

The format of the <method> element for creating or replacing a CGI script is:

```
<method methodName="cgi.createOrReplaceCGI">
```

Use this method to register a new CGI script with the Web GUI server or replacing the existing definition if it already exists. The <method> element contains one or more <cgi> elements each defining a script to register. For more information, see "<cgi>" on page 66.

This example creates or replaces a script named myping1.

```
<methodCall>
 <method methodName="cgi.createOrReplaceCGI">
  <cgi name="myping1" acl="*" useSmartPageCommands="false"
   fileName="/home/nco_myping1.cgi">
  </cgi>
 </method>
</methodCall>
```

Modify a CGI script

The format of the <method> element for modifying a CGI script is:

```
<method methodName="cgi.modifyCGI">
```

Use this method to modify a previously registered CGI script. The <method> element contains one or more <cgi> elements each modifying the characteristics of a CGI script. For more information, see "<cgi>" on page 66.

Example

This example changes the value of the useSmarypage attribute for the script named myping.

```
<methodCall>
 <method methodName="cgi.modifyCGI">
  <cgi name="myping" acl="example" useSmartPageCommands="false" >
  </cgi>
 </method>
</methodCall>
```

Unregister a CGI script

The format of the <method> element for unregistering a CGI script is:

```
<method methodName="cgi.unregisterCGI">
```

Use this method to unregister a previously registered CGI script. The <method> element contains one or more <cgi> elements each identifying a CGI script to unregister. In the <cgi> element, include only the name attribute.

Example

This example unregisters a script named myping.

```
<methodCall>
 <method methodName="cgi.unregisterCGI">
  <cgi name="myping" >
  </cgi>
 </method>
</methodCall>
```

Filter requests

Filter requests operate on filters used to display events in the Active Event List (AEL), the Event Viewer, Lightweight Event List (LEL), Table View, and on monitor boxes in the Event Dashboard. There are functions to create, modify, remove, and list filters. you can also set the default view. WAAPI provides six methods for operating on filters: adding a filter, creating or replacing a filter, modifying a filter, deleting a filter, getting a list of filters, and setting the default view for a filter.

Add a filter

The format of the `<method>` element for adding a filter is:

```
<method methodCall="filter.addFilter">
```

Use this method to define a new filter for use on the AEL, Event Viewer, LEL, Table View, and the Event Dashboard. The `<method>` element contains one or more `<filter>` elements, each of which defines the characteristics of a new filter.

For a dependent filter, the `<filter>` element contains one or more instances of the `<dependentlist>` element.

- "<filter>"
- "<dependentlist>" on page 70
- "Rules for dependent filters" on page 70
- "Example: Event filter" on page 71
- "Example: Group filter" on page 71
- "Example: Dependent filter" on page 72

<filter>

The `<filter>` element defines the characteristics of a filter and has the following attributes:

Table 32. Attributes of the <filter> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a filter. Value: String Default value: None
user	Optional	For user filters, this attribute identifies the users that the filter is associated with. The value of the attribute is a comma-separated list of User IDs. Value: List of User IDs Default value: None
group	Optional	For group filters, this attribute identifies the groups that the filter is associated with. The value of the attribute is a comma-separated list of Group IDs. Value: List of Group IDs Default value: None

Table 32. Attributes of the <filter> element (continued)

Attribute name	Required or optional	Description
sql	Optional	The value of the SQL WHERE clause that defines the fields to use and their values when filtering events. Value: A SQL WHERE clause Default value: None
mode	Optional	Defines whether the filter is a dependent filter. Include this attribute for dependent filters only. Value: `dependent` Default value: None
description	Optional	Defines a textual description of the filter. Value: String Default value: None
filtercollection	Optional	Specifies the name of a filter collection that the filter is a member of. Value: String Default value: None
type	Optional	Identifies the type of the filter. Value: `global`, `system`, `user`, or `group` Default value: None
view	Optional	Identifies the view associated with the filter. Value: String Default value: None
viewtype	Optional	Identifies the type of view. Value: `global`, `system`, or `user` Default value: None
datasource	Optional	Identifies the data source that supplies events for filtering. To specify more than one data source use a comma-separated list. This attribute is required when adding a filter. Value: String Default value: None
metriclabel	Optional	Defines the label to describe the aggregate statistic data displayed on monitor boxes in the Event Dashboard. Value: String Default value: `Metric:`

Table 32. Attributes of the <filter> element (continued)

Attribute name	Required or optional	Description
metricshow	Optional	Defines the calculation criteria for the metric value displayed on monitor boxes in the Event Dashboard. Value: `Average`, `Count`, `Sum`, `Minimum`, or `Maximum` Default value: `Average`
metricof	Optional	Defines the ObjectServer field to use in calculating the metric value displayed on monitor boxes in the Event Dashboard. Value: Field name Default value: `Severity`

<dependentlist>

The `<dependentlist>` element identifies the list of filters that a dependent filter uses and has the following attributes.

Table 33. Attributes of the <dependentList> element

Attribute name	Required or optional	Description
type	Required	Identifies the type of filters in this list of dependent filters. Value: `global`, `system`, `group`, or `user` Default value: None
list	Required	Defines the filters associated with a dependent filter. The attribute's value contains a comma-separated list of one or more filter names. Value: List of filter names Default value: None

Rules for dependent filters

Each type of dependent filter, as determined by the value of the `type` attribute in the `<dependentList>` element, can depend on certain types of filter only:

Table 34. Rules for the types of filter that a dependent filter can depend on

Type of dependent filter	Rules
Global System	A global or system dependent filter: • Can depend on system and global filters • Cannot depend on any user filter
User	A user dependent filter: • Can depend on global and system filters • Can depend on other user filters for the same user

Table 34. Rules for the types of filter that a dependent filter can depend on (continued)

Type of dependent filter	Rules
Group	A group dependent filter: • Can depend on global and system filters • Can depend on other group filters for the same user group • Cannot depend on any user filter

Example: Event filter

This example adds a filter named `exampleFilter1` that has the following characteristics:

- The SQL WHERE clause is `Severity>=0`
- The metric label is `Metric:`
- The name of the view associated with the filter is `advanced`
- The filter is a global filter
- The name of the data source is `NCOMS`

```
<methodCall>
 <method methodName="filter.addFilter">
  <filter name="exampleFilter1"
   sql="Severity >=0"
   metriclabel="Metric : "
   view="advanced"
   type="global"
   datasource="NCOMS"
  />
 </method>
</methodCall>
```

Example: Group filter

This example adds a group filter named `exampleGroupFilter` that has the following characteristics:

- The SQL WHERE clause is `Severity>=0`
- The metric label is `Metric:`
- The name of the view associated with the filter is `advanced` and that is a `global` view
- The filter is available to users in the groups `Netcool_OMNIbus_Admin` and `Netcool_OMNIbus_User`
- The name of the data source is `NCOMS`

```
<methodCall>
 <method methodName="filter.addFilter">
  <filter name="exampleGroupFilter"
   sql="Severity >=0"
   metriclabel="Metric : "
   view="advanced"
   viewtype="global"
   type="group"
   group="Netcool_OMNIbus_Admin,Netcool_OMNIbus_User"
   datasource="NCOMS"/>
 </method>
</methodCall>
```

Example: Dependent filter

This example adds a dependent filter named `exampleFilter3` that has the following characteristics:

- The metric label is `Metric:`
- The name of the view associated with the filter is `advanced`
- The filter is a global filter
- The name of the data source is `NCOMS`
- The filter is dependent on the filters named `dependentfilter1` and `dependentfilter2`

```
<methodCall>
 <method methodName="filter.addFilter">
  <filter name="exampleFilter3"
   metriclabel="Metric : "
   view="advanced"
   type="global"
   mode="dependent"
   datasource="NCOMS">
   <dependentlist type="global" list="dependentFilter1,dependentFilter2"/>
  </filter>
 </method>
</methodCall>
```

Create or replace a filter

The format of the `<method>` element for creating or replacing a filter is:

```
<method methodName="filter.createOrReplaceFilter">
```

Use this method to create a new filter or replace one that already exists. The `<method>` element contains one or more `<filter>` elements each of which defines the characteristics of a filter. For more information, see "<filter>" on page 68.

For a dependent filter, the `<filter>` element contains one or more instances of the `<dependentlist>` element. For more information, see "<dependentlist>" on page 70.

This example creates or replaces a filter named `exampleFilter2` that has the following characteristics:

- The SQL WHERE clause is `Severity>=1`
- The metric label is `My Metric:`
- The name of the view associated with the filter is `advanced`
- The filter is a global filter
- The name of the data source is `NCOMS`

```
<methodCall>
 <method methodName="filter.createOrReplaceFilter">
  <filter name="exampleFilter2"
   sql="Severity >=1"
   metriclabel="My Metric : "
   view="advanced"
   type="global"
   datasource="NCOMS"
  />
 </method>
</methodCall>
```

Delete a filter

The format of the <method> element for deleting a filter is:

```
<method methodName="filter.deleteFilter">
```

Use this method to delete an existing filter. The <method> element contains one or more <filter> elements, each of which identifies a filter to delete. In the <filter> element, include the name and type attributes. In addition, for user filters, include the user attribute to define the users from whom the filter is to be removed. Similarly for group filters, use the group attribute to define the groups from which the filter is to be removed. For more information, see "<filter>" on page 68.

Example

This example deletes the global filters name exampleFilter1, exampleFilter3, the user filter named exampleFilter2, and the global, dependent filters named dependentFilter1 and dependentFilter2. The user filter is removed from the User IDs webtopadminuser and root.

```
<methodCall>
 <method methodName="filter.deleteFilter">
  <filter name="exampleFilter1" type="global"/>
  <filter name="exampleFilter2" type="user" user="webtopadminuser,root"/>
  <filter name="exampleFilter3" type="global"/>
  <filter name="dependentFilter1" type="global"/>
  <filter name="dependentFilter2" type="global"/>
 </method>
</methodCall>
```

Get a list of filters

The format of the <method> element for getting a list of filters is:

```
<method methodName="filter.getList" />
```

Use this method to obtain a list containing the names of the defined filters. The list contains a separate section for each type of filter (global, system, user, and group).

This method does not return the attributes of the filters, only the names.

Example

```
<methodCall>
 <method methodName="filter.getList" />
</methodCall>
```

Modify a filter

The format of the <method> element for modifying a filter is:

```
<method methodName="filter.modify">
```

Use this method to modify the characteristics of an existing filter. The <method> element contains one or more <filter> elements each of which defines the new characteristics of an existing filter. Include only attributes of the <filter> that correspond to the characteristics you want to change. When you omit an attribute the corresponding characteristic is unchanged. For more information, see "<filter>" on page 68.

For a dependent filter, the <filter> element contains one or more instances of the <dependentlist> element. For more information, see "<dependentlist>" on page 70.

Example

This example modifies the filter named exampleFilter1 and makes the following changes to the filter's characteristics:

- The SQL WHERE clause becomes lastOccurrence >= getdate-1800
- The metric label becomes MyMetric:

In addition, the example adds the following characteristics to the filter:

- The view is a global view
- The metric value is the average of the SubDivision field

```
<methodCall>
 <method methodName="filter.modifyFilter">
  <filter name="exampleFilter1"
   sql="LastOccurrence >= getdate -1800"
   metriclabel="MyMetric:"
   metricshow="Average"
   metricof="SubDivision"
   view = "advanced"
   viewtype="global"
   type="global"
   datasource="NCOMS"
  />
 </method>
</methodCall>
```

Set the default view

The format of the <method> element for setting the default view is:

```
<method methodName="filter.setDefaultView">
```

Use this method to determine the default view associated with a filter. The <method> element contains one or more <filter> elements each of which identifies a filter to associate a default view with. In the <filter> element, include the name to identify the filter and the view and viewtype attributes to identify default view for the filter. For more information, see "<filter>" on page 68.

Example

This example associates the global view named Default with the filter named exampleFilter1:

```
<methodCall>
 <method methodName="filter.setDefaultView">
  <filter name="exampleFilter1" view="Default" viewtype="global"/>
 </method>
</methodCall>
```

Filter collection requests

Filter collections are logical groupings of filters that are typically used for entity data migrated from Netcool/Webtop. There are functions to create, modify, delete, and list filter collections. In addition, you can add and delete filters from a collection, get a list of file collections, and set the default view.

WAAPI provides eight methods for operating on filter collections: creating a filter collection, creating or replacing a filter collection, modifying a filter collection, deleting a filter collection, adding a filter to a filter collection, deleting a filter from a filter collection, getting a list of filter collections, and setting the view for a filter collection.

Create a filter collection

The format of the <method> element for creating a filter collection is:

```
<method methodCall="filtercollection.createFilterCollection">
```

Use this method to define a new collection of filters. The <method> element contains one or more <filterCollection> elements each of which defines the characteristics of a new collection. Each of these elements contains one or more <filter> elements that identify filters to be part of the collection. You can specify only system and global filters that already exist in the system. The <filter> elements contain use only the name and type attributes to identify the filter. For more information, see "<filter>" on page 68.

- "<filterCollection>"
- "Example"

<filterCollection>

The <filterCollection> element defines the characteristics of a filter collection. the element contains one or more <filter> elements that identify the filters in the collection, using only the name and type attributes, and has the following attributes:

Table 35. Attributes of the <filterCollection> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a filter collection. Value: String Default value: None
description	Optional	Defines a textual description of the filter collection. Value: String Default value: None
viewName	Optional	Identifies the view associated with the filter. Value: String Default value: None
viewType	Optional	Identifies the type of view. Value: global or system Default value: global

Example

The following example creates a filter collection named testcollection that contains two filters named Example_Critical and AllEvents.

```
<methodCall>
 <method methodName="filtercollection.createFilterCollection">
  <filterCollection name="testcollection" viewName="Default"
   viewType="global" description="A collection of filters for testing">
   <filter name="Example_Critical" type="system"/>
```

```
   <filter name="AllEvents" type="global"/>
  </filterCollection>
 </method>
</methodCall>
```

Add a filter to a filter collection

The format of the <method> element for adding a filter to a collection is:

```
<method methodCall="filtercollection.addFilter">
```

Use this method to add a filter to an existing collection of filters. The <method> element contains one or more <filterCollection> elements each of which identifies a collection. For more information, see "<filterCollection>" on page 75. Each of these elements contains one or more <filter> elements that identify filters to add to the collection. You can specify only system and global filters that already exist in the system. The <filter> elements contain use only the name and type attributes to identify the filter. For more information, see "<filter>" on page 68.

Example

The following example adds filters named Last10Mins and Information to the collection named testcollection2.

```
<methodCall>
 <method methodName="filtercollection.addFilter">
  <filterCollection name="testcollection2">
   <filter name="Last10Mins" type="global"/>
   <filter name="Information" type="global"/>
  </filterCollection>
 </method>
</methodCall>
```

Create or replace a filter collection

The format of the <method> element for creating or replacing a filter collection is:

```
<method methodCall="filtercollection.createOrReplaceFilterCollection">
```

Use this method to define a new collection of filters or replace one that already exists. The <method> element contains one or more <filterCollection> elements each of which defines the characteristics of a collection. For more information, see "<filterCollection>" on page 75. Each of these elements contains one or more <filter> elements that identify filters to be part of the collection. You can specify only system and global filters that already exist in the system. The <filter> elements contain use only the name and type attributes to identify the filter. For more information, see "<filter>" on page 68.

Example

The following example creates or replaces a filter collection named testcollection2.

```
<methodCall>
 <method methodName="filtercollection.createOrReplaceFilterCollection">
  <filterCollection name="testcollection2" viewName="Default"
   viewType="global" description="Another collection of filters for testing">
   <filter name="Escalated" type="global"/>
   <filter name="AllEvents" type="global"/>
  </filterCollection>
 </method>
</methodCall>
```

Delete a filter collection

The format of the <method> element for deleting a filter collection is:

```
<method methodCall="filtercollection.deleteFilterCollection">
```

Use this method to delete an existing collection of filters. The <method> element contains one or more <filterCollection> elements each of which identifies a collection to delete. In the <filterCollection> element include only the name attribute. For more information, see "<filterCollection>" on page 75.

Example

The following example deletes the filter collection named testCollection.

```
<methodCall>
 <method methodName="filtercollection.deleteFilterCollection">
  <filterCollection name="testcollection"/>
 </method>
</methodCall>
```

Delete a filter from a filter collection

The format of the <method> element for deleting a filter from a collection is:

```
<method methodCall="filtercollection.deleteFilter">
```

Use this method to delete a filter to an existing collection of filters. The <method> element contains one or more <filterCollection> elements each of which identifies a collection. For more information, see "<filterCollection>" on page 75. Each of these elements contains one or more <filter> elements that identify filters to delete from the collection. The <filter> elements contain use only the name and type attributes to identify the filters. For more information, see "<filter>" on page 68.

Example

The following example removes the filter named Information from the collection named testcollection2

```
<methodCall>
 <method methodName="filtercollection.deleteFilter">
  <filterCollection name="testcollection2">
   <filter name="Information" type="global"/>
  </filterCollection>
 </method>
</methodCall>
```

Get a list of filter collections

The format of the <method> element for getting a list of filter collections is:

```
<method methodCall="filtercollection.getList">
```

Use this method to obtain a list of defined filter collections.

Example

```
<methodCall>
 <method methodName="filtercollection.getList />
</methodCall>
```

Modify a filter collection

The format of the <method> element for modifying a filter collection is:

```
<method methodCall="filtercollection.modifyFilterCollection">
```

Use this method to modify the characteristics of an existing collection of filters. The <method> element contains one or more <filterCollection> elements each of which defines the characteristics of a collection. Each of these elements contains one or more <filter> elements that identify filters to be part of the collection. You can specify only system and global filters that already exist in the system. The <filter> elements contain use only the name and type attributes to identify the filter.

Example

The following example modifies the filter collection named testCollection2.

```
<methodCall>
 <method methodName="filtercollection.modifyFilterCollection">
  <filterCollection name="testcollection2" viewName="Default" viewType="global"
   description="A new description for testcollection2">
   <filter name="TaskList" type="global"/>
   <filter name="NetcoolStatus" type="global"/>
  </filterCollection>
 </method>
</methodCall>
```

Set the view for a filter collection

The format of the <method> element for setting the default view of a collection is:

```
<method methodName="filtercollection.setView">
```

Use this method to set the view associated with a filter. The <method> element contains one or more <filterCollection> elements each of which identifies a collection to associate a default view with. In the <filterCollection> element, include the name to identify the filter and the viewName and viewType attributes to identify the view for the filter. For more information, see “<filterCollection>” on page 75.

Example

The following example sets the view for the filter collection named testcollection2.

```
<methodCall>
 <method methodName="filtercollection.setView">
  <filterCollection name="testcollection2" viewName="DefaultTable"
   viewType="global"/>
 </method>
</methodCall>
```

Metric requests

Metric requests operate on performance metrics that the Web GUI displays in the form of gauges. There are functions to create, modify, replace, and delete metrics. You can also obtain a list of the currently defined metrics. WAAPI provides five methods for working on metrics: Creating a metric, creating or replacing a metric, modifying a metric, deleting a metric, and getting a list of metrics.

Create a metric

The format of the <method> element for creating a metric is:

```
<method methodName="metric.create">
```

Use this method to define a new metric for use on a gauge. The <method> element contains one or more <metric:metric> elements each defining the characteristics of a new metric.

Each <metric:metric> element contains one instance of the <metric:command> element. In turn, the <metric:command> element contains a <metric:text> element.

- "<metric:metric>"
- "<metric:command>" on page 81
- "<metric:text>" on page 81
- "Examples" on page 82

<metric:metric>

The <metric:metric> element defines the characteristics of a metric and has the following attributes:

Table 36. Attributes of the <metric:metric> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a metric. Value: String Default value: None
description	Optional	Defines a textual description of the metric that appears when the user hovers the mouse pointer over the gauge. To include the current value of the metric in the description, use {0} at the appropriate place. Value: String Default value: None
descriptionKey	Optional	A key for the description attribute that uniquely identifies it among all defined metrics. Omit this attribute to allow the system to automatically generate a unique key for you. Value: String Default value: A system-generated value.

Table 36. Attributes of the <metric:metric> element (continued)

Attribute name	Required or optional	Description
displayName	Optional	The name of the metric as it appears beneath a gauge on the Gauges page. Value: String Default value: None
displayNameKey	Optional	A key for the `displayName` attribute that uniquely identifies it among all defined metrics. Omit this attribute to allow the system to automatically generate a unique key for you. Value: String Default value: A system-generated value
units	Optional	Defines the units that the metric displays. For example the number of incidents that have occurred, or the number of client connections to a server. Value: String Default value: None
unitsKey	Optional	A key for the units attribute that uniquely identifies it among all defined metrics. Omit this attribute to allow the system to generate a unique key for you. Value: String Default value: A system-generated value
maxValue	Optional	Defines the maximum value that the metric can have, expressed in terms of the metric's units. Value: Integer Default value: None
minValue	Optional	Defines the minimum value that the metric can have, expressed in terms of the metric's units. Value: Integer Default value: None
threshold1	Optional	Defines the threshold between the low and medium ranges of the metric. The attribute defines the threshold as a percentage of the metrics value range and its value must be lower than `threshold2`. Value: Integer Default value: None

Table 36. Attributes of the <metric:metric> element (continued)

Attribute name	Required or optional	Description
threshold2	Optional	Defines the threshold between the medium and high ranges of the metric. The attribute defines the threshold as a percentage of the metric's value range and its value must be higher than `threshold1`. Value: Integer Default value: None

Each `<metric::metric>` can contain one instance of `<metric:command>`.

<metric:command>

The `<metric:command>` element defines the command that generates values for the metric. Typically this is a SQL command that queries the ObjectServer database. The element can have the following attribute:

Table 37. Attributes of the <metric:command> element

Attribute name	Required or optional	Description
type	Optional	Defines the type of command that generates values for the metric. Currently the only valid value for this attribute is `sql`. Value: String Default value: `sql`
mode	Optional	Defines the mode to use when the metric is in use. In basic mode, any restriction filters active on the specified database tables are applied. In advanced mode any restriction filters are not applied when the user accesses the metric using a gauge. Value: `basic` or `advanced` Default value: `advanced`

If present, the `<metric:command>` element contains one instance of the `<metric:text>` element.

<metric:text>

The `<metric:text>` element contains the command that generates values for the metric. The element has the following attribute:

Table 38. Attributes of the <metric:text> element

Attribute name	Required or optional	Description
data	See Description	Contains the text of the command used to generate values for the metric. This attribute is required when the `mode` attribute of the `<metric:command>` element is omitted or has a value of `advanced`. Value: String Default value: None
selectField	See Description	Contains the table field or an aggregate function of the SELECT clause of the command used to generate values for the metric. This attribute is required when the `mode` attribute of the `<metric:command>` element has a value of `basic`. Value: String Default value: None
whereClause	See Description	Contains the WHERE clause of the command used to generate values for the metric. This attribute is required when the `mode` attribute of the `<metric:command>` element has a value of `basic`. Value: String Default value: None
databaseName	See Description	Contains the database name clause of the command used to generate values for the metric. This attribute is required when the `mode` attribute of the `<metric:command>` element has a value of `basic`. Value: String Default value: None
tableName	See Description	Contains the table name clause of the command used to generate values for the metric. This attribute is required when the `mode` attribute of the `<metric:command>` element has a value of `basic`. Value: String Default value: None

Examples

The following example creates a metric named `metricsample1` in advanced mode. The metric measures the number of critical events outstanding and has the following characteristics:

- A minimum value of 0 and a maximum of 10000

- A low to medium threshold of 30%
- A medium to high threshold of 70%

```
<methodCall xmlns:metric="http://www.ibm.com/tivoli/netcool/webtop/metrics/7.4.0">
 <method methodName="metric.createMetric">
  <metric:metric name="metricsample1"
   displayName="MetricSample1"
   description="Critical events. Current value: {0}"
   units="events"
   maxValue="10000"
   minValue="0"
   threshold1="30"
   threshold2="70">
   <metric:command type="sql">
    <metric:text data="select sum(Tally) from alerts.status where Severity=5;"/>
   </metric:command>
  </metric:metric>
 </method>
</methodCall>
```

The following example creates a similar metric in basic mode. So any restriction filters that are active when the user accesses the metric are applied.

```
<methodCall>
 <method methodName="metric.createMetric">
  <metric:metric name="metricsample1"
   displayName="MetricSample1"
   description="Critical events. Current value {0}"
   units="events"
   maxValue="10000"
   minValue="0"
   threshold1="30"
   threshold2="70">
   <metric:command type="sql" mode="basic">
    <metric:text selectField="sum(Tally)" whereClause="Severity=5"
    databaseName="alerts" tableName="status"/>
   </metric:command>
  </metric:metric>
 </method>
</methoodCall>
```

Create or replace a metric

The format of the <method> element for creating or replacing a metric is:

```
<method methodName="metric.createOrReplaceMetric">
```

Use this method to replace an existing metric or create a new one if it does not already exist. The <method> element contains one or more <metric:metric> elements each of which defines the characteristics of a new metric. Each <metric:metric> element contains a <metric:command> element which, in turn, contains a <metric:text> element. For more information, see the following descriptions of these elements:

- "<metric:metric>" on page 79
- "<metric:command>" on page 81
- "<metric:text>" on page 81

Example

The following example creates or replaces a metric named metricsample2 in advanced mode. The new metric measures the number of major events outstanding and has the following characteristics:

- A minimum value of 0

- A maximum value of 100
- A low to medium threshold of 40%
- A medium to high threshold of 80%

```
<methodCall xmlns:metric="http://www.ibm.com/tivoli/netcool/webtop/metrics/7.4.0">
 <method methodName="metric.createOrReplaceMetric">
  <metric:metric name="metricsample2"
   displayName="MetricSample2"
   description="Major events. Current value: {0}"
   units="events"
   maxValue="100"
   minValue="0"
   threshold1="40"
   threshold2="80">
   <metric:command type="sql">
    <metric:text data="select sum(Tally) from alerts.status where Severity=4;"/>
   </metric:command>
  </metric:metric>
 </method>
</methodCall>
```

To create or replace a metric in basic mode, use the same format of the <metric:command> and <metric:text> elements shown in the example of creating a metric in basic mode.

Delete a metric

The format of the <method> element for deleting a metric is:

```
<method methodName="metric.deleteMetric">
```

Use this method to delete an existing metric. The <method> element contains one or more <metric:metric> elements each of which identifies a metric to delete. In the <metric:metric> element include only the name attribute. For more information, see "<metric:metric>" on page 79.

Example

This example deletes the metric named metricsample2.

```
<methodCall xmlns:metric="http://www.ibm.com/tivoli/netcool/webtop/metrics/7.4.0">
 <method methodName="metric.deleteMetric">
  <metric:metric name="metricsample2"/>
 </method>
</methodCall>
```

Get a list of metrics

The format of the <method> element for getting a list of metrics is:

```
<method methodName="metric.getList" />
```

Use this method to obtain a list containing the names of the defined metrics.

Example

```
<methodCall xmlns:metric="http://www.ibm.com/tivoli/netcool/webtop/metrics/7.4.0">
 <method methodName="metric.getList" />
</methodCall>
```

Modify a metric

The format of the <method> element for modifying a metric is:

```
<method methodName="metric.modifyMetric">
```

Use this method to modify the characteristics of an existing metric. The <method> element contains one or more <metric:metric> elements each of which defines the new characteristics of an existing metric. Include only attributes of the <metric:metric> that correspond to the characteristics you want to change. When you omit an attribute the corresponding characteristic is unchanged.

To modify the command associated with the metric, include a <metric:command> element which, in turn, contains the <metric:text> element.

For more information, see the following descriptions of these elements:

- "<metric:metric>" on page 79
- "<metric:command>" on page 81
- "<metric:text>" on page 81

Example

This example modifies the metric named metricsample1 and makes the following changes to the metric's characteristics:

- The text of the description is enhanced
- The maximum value changes from 10000 to 250
- The low to medium threshold changes from 30% to 40%
- The medium to high threshold changes from 70% to 90%

```
<methodCall xmlns:metric="http://www.ibm.com/tivoli/netcool/webtop/metrics/7.4.0">
 <method methodName="metric.modifyMetric">
  <metric:metric name="metricsample1"
   displayName="MetricSample1"
   description="Critical events. Modified by WAAPI. Current value: {0}"
   units="events"
   maxValue="250"
   minValue="0"
   threshold1="40"
   threshold2="90">
   <metric:command type="sql">
    <metric:text data="select sum(Tally) from alerts.status where Severity=5;"/>
   </metric:command>
  </metric:metric>
 </method>
</methodCall>
```

To modify a metric in basic mode, use the same format of the <metric:command> and <metric:text> elements shown in the example of creating a metric in basic mode.

Relationship requests

Relationship requests operate on event relationships that can enhance the display of events in the Event Viewer. There are functions to create, modify, replace and delete relationships. You can also obtain a list of the currently defined relationships. WAAPI provides five methods for working on relationships: Creating a relationship, creating or replacing a relationship, modifying a relationship, deleting a relationship, and getting a list of relationships.

Create a relationship

The format of the <method> element for creating a relationship is:

```
<method methodName="relationship.createRelationship">
```

Use this method to create a new relationship for use in the Event Viewer. The <method> element contains one or more <relationship:relationship> elements, each defining the characteristics of a new relationship.

Each <relationship:relationship> element contains one instance of the <relationship:definition> element. In turn, the <relationship:definition> element contains one instance of the <relationship:column> element.

- "<relationship:relationship>"
- "<relationship:definition>" on page 87
- "<relationship:column>" on page 88
- "Example" on page 88

<relationship:relationship>

The <relationship:relationship> element defines the characteristics of a relationship and has the following attributes:

Table 39. Attributes of the <relationship:relationship> element

Attribute name	Required or optional	Description
name	Required	Provides a unique name for a relationship. Value: String Default value: None
description	Optional	Defines a textual description of the relationship. This appears in the Relationship Editor in the Web GUI. Value: String Default value: None
descriptionKey	Optional	A key for the description attribute that uniquely identifies it among all defined relationships. Omit this attribute to allow the system to automatically generate a unique key for you. Value: String Default value: A system-generated value.

Table 39. Attributes of the <relationship:relationship> element (continued)

Attribute name	Required or optional	Description
displayName	Required	The name of the metric as it appears in the Relationship Editor. Value: String Default value: None
displayNameKey	Optional	A key for the `displayName` attribute that uniquely identifies it among all defined relationships. You can omit this attribute to allow the system to automatically generate a unique key for you. However, including this attribute helps to implement globalization of your application, where this value identifies the localized version of the display name in various languages. Value: String Default value: A system-generated value
dataSources	Optional	The name of the data source that provides events to be grouped according to this relationship. To specify more than one data source, use a comma-separated list. Value: String Default value: None

Each `<relationship:relationship>` element contains one instance of the `<relationship:definition>` element.

<relationship:definition>

The `<relationship:definition>` element defines a relationship between two fields in the ObjectServer and has the following attribute:

Table 40. Attributes of the <relationship:definition> element

Attribute name	Required or optional	Description
type	Required	Defines the type of relationship. Currently the only valid value for this attribute is `relationshipColumn`. Value: String Default value: `relationshipColumn`

If present, the `<relationship:definition>` element contains one instance of the `<relationship:column>` element.

<relationship:column>

The <relationship:column> element defines two ObjectServer columns that make up an event relationship and has the following attributes:

Table 41. Attributes of the <metric:command> element

Attribute name	Required or optional	Description
column	Required	Defines the column that defines the relationship. Value: String Default value: None
keyColumn	Required	Defines the name of the column that is the key for the column defined in the column attribute. Value: String Default value: None

Example

This example adds a relationship named SummaryRelationship1 that has the following characteristics:

- This example adds a relationship named SummaryRelationship1 that has the following characteristics:
  - Uses the column named NmosSerial
  - Is keyed off the Summary column

```
<methodCall xmlns:relationship="http://www.ibm.com/tivoli/netcool/
webtop/relationships/7.4">
 <method methodName="relationship.createRelationship">
  <relationship:relationship name="SummaryRelationship1"
   displayName="RelationshipSample1"
   description="An example relationship created with WAAPI.">
   <relationship:definition type="relationshipColumn">
    <relationship:column column="NmosSerial" keyColumn="Summary"/>
   </relationship:definition>
  </relationship:relationship>
 </method>
</methodCall>
```

Create or replace a relationship

The format of the <method> element for creating or replacing a relationship is:

```
<method methodName="relationship.createOrReplaceRelationship">
```

Use this method to replace an existing relationship or to create a new one if it does not already exist. The <method> element contains one or more <relationship:relationship> elements, each of which defines the characteristics of a relationship. Each <relationship:relationship> contains one <relationship:definition> element which in turn contains a <relationship:column> element. For more information about the <relationship:relationship> element and its subelements, see "<relationship:relationship>" on page 86.

The following example creates or replaces a relationship named SummaryRelationship2.

```
<methodCall xmlns:relationship="http://www.ibm.com/tivoli/netcool/
webtop/relationships/7.4">
 <method methodName="relationship.createOrReplaceRelationship">
  <relationship:relationship name="SummaryRelationship2"
   displayName="RelationshipSample1"
   description="An example relationship created with WAAPI.">
   <relationship:definition type="relationshipColumn">
    <relationship:column column="ParentSerial" keyColumn="Serial"/>
   </relationship:definition>
  </relationship:relationship>
 </method>
</methodCall>
```

Delete a relationship

The format of the <method> element for deleting a relationship is:

```
<method methodName="relationship.deleteRelationship">
```

Use this method to delete an existing relationship. The <method> element contains one or more <relationship:relationship> elements, each of which identifies a relationship to delete. In the <relationship:relationship> element, include only the name attribute. For more information about the <relationship:relationship> element and its subelements, see "<relationship:relationship>" on page 86.

The following example deletes a relationship named SummaryRelationship2.

```
<methodCall xmlns:relationship="http://www.ibm.com/tivoli/netcool/
webtop/relationships/7.4">
 <method methodName="relationship.deleteRelationship">
  <relationship:relationship name="SummaryRelationship2"
  </relationship:relationship>
 </method>
</methodCall>
```

Get a list of relationships

The format of the <method> element for getting a list of relationships is:

```
<method methodName="relationship.getList">
```

Use this method to obtain a list containing the names of the defined relationships.

```
<methodCall xmlns:relationship="http://www.ibm.com/tivoli/netcool/
webtop/relationships/7.4">
 <method methodName="relationship.getList" />
</methodCall>
```

Modify a relationship

The format of the <method> element for modifying a relationship is:

```
<method methodName="relationship.modifyRelationship">
```

Use this method to modify the characteristics of an existing relationship. The <method> element contains one or more <relationship:relationship> elements, each of which defines the new characteristics of an existing relationship. Include only attributes of the <relationship:relationship> element that you want to change. When you omit an attribute, the corresponding characteristic is unchanged.

The <relationship:rrelationship> element contains zero or one <relationship:definition> elements. Include that element only when you want to modify the columns associated with a relationship. If present, the <relationship:definition> element contains one <relationship:column> element.

For more information about the <relationship:relationship> element and its subelements, see "<relationship:relationship>" on page 86.

The following example modifies a relationship named SummaryRelationship1, changing the description:

```
<methodCall xmlns:relationship=
"http://www.ibm.com/tivoli/netcool/webtop/relationships/7.4">
 <method methodName="relationship.modifyRelationship">
  <relationship:relationship name="SummaryRelationship1"
   description="Relationship based on the Summary field">
  </relationship:relationship>
 </method>
</methodCall>
```

Other requests

There are miscellaneous functions to resynchronize the Web GUI cache, to remove a node from a cluster, to generate a status report, and to reload the server's filters and views.

WAAPI provides four other functions:

- "Resynchronize the Web GUI cache with the ObjectServer's database"
- "Remove a node from a load balancing cluster"
- "Generate a system status report" on page 91
- "Reload filters and views" on page 91

Resynchronize the Web GUI cache with the ObjectServer's database

The format of the <method> element for resynchronizing the Web GUI cache with the ObjectServer's database is:

```
<method methodName="osresync.refreshOSCache" />
```

Use this method to force the Web GUI to resynchronize its cache with the ObjectServer's database. This function is especially useful when a user's ObjectServer permissions have changed and it ensures that the Web GUI has the latest information and so can manage the user correctly.

Example

```
<methodCall>
 <method methodName="osresync.refreshOSCache" />
</methodCall>
```

Remove a node from a load balancing cluster

The format of the <method> element for removing a node from a cluster is:

```
<method methodName="cluster.removeNode">
```

Use this method to remove a node from a load balancing cluster. The method removes the node that you send this request to. If this is the last node in the cluster, the method also removes all the load balancing configuration data.

Example

```
<methodCall>
 <method methodName="cluster.removeNode" />
</methodCall>
```

Generate a system status report

The format of the <method> element for generating a system status report is:

```
<method methodName="webtopprobe.generateReport">
```

Use this method to generate a status report for the Web GUI. The generated report contains the same information as the **Troubleshooting and support** > **System Information for Tivoli Netcool/OMNIbus Web GUI** function of the Web GUI generates and contains the following sections:

- Version numbers of the Web GUI, the Dashboard Common Infrastructure (DCI), and Java Runtime Environment
- Memory usage statistics
- Runtime platform information
- Protocols in use
- ObjectServer properties and configuration data, including information about the cache
- All system properties, including those internal to the Web GUI

Example

```
<methodCall>
 <method methodName="webtopprobe.generateReport" />
<methodCall>
```

Reload filters and views

The format of the <method> element to reload filters and views is:

```
<method methodName="xmldao.reloadFiltersAndViews">
```

Use this method to force the Web GUI to reload all currently active filters and views. In addition, this method forces AEL clients to update their filters and views during the next refresh.

Example

```
<methodCall>
 <method methodName="xmldao.reloadFiltersAndViews" />
</methodCall>
```

Generate a system status report

The format of the <method> element for generating a system status report is:

```
<method methodName="webtopprobe.generateReport">
```

Use this method to generate a status report for the Web GUI. The generated report contains the same information as the **Troubleshooting and support** > **System Information for Tivoli Netcool/OMNIbus Web GUI** function of the Web GUI generates and contains the following sections:

- Version numbers of the Web GUI, the Dashboard Common Infrastructure (DCI), and Java Runtime Environment
- Memory usage statistics
- Runtime platform information
- Protocols in use
- ObjectServer properties and configuration data, including information about the cache
- All system properties, including those internal to the Web GUI

Example

```
<methodCall>
 <method methodName="webtopprobe.generateReport" />
</methodCall>
```

Reload filters and views

The format of the <method> element to reload filters and views is:

```
<method methodName="xmldao.reloadFiltersAndViews">
```

Use this method to force the Web GUI to reload all currently active filters and views. In addition, this method forces AEL clients to update their filters and views during the next refresh.

Example

```
<methodCall>
 <method methodName="xmldao.reloadFiltersAndViews" />
</methodCall>
```

Remove a node from a load balancing cluster

The format of the <method> element for removing a node from a cluster is:

```
<method methodName="cluster.removeNode">
```

Use this method to remove a node from a load balancing cluster. The method removes the node that you send this request to. If this is the last node in the cluster, the method also removes all the load balancing configuration data.

Example

```
<methodCall>
 <method methodName="cluster.removeNode" />
</methodCall>
```

Resynchronize the Web GUI cache with the ObjectServer's database

The format of the <method> element for resynchronizing the Web GUI cache with the ObjectServer's database is:

```
<method methodName="osresync.refreshOSCache" />
```

Use this method to force the Web GUI to resynchronize its cache with the ObjectServer's database. This function is especially useful when a user's ObjectServer permissions have changed and it ensures that the Web GUI has the latest information and so can manage the user correctly.

Example

```
<methodCall>
 <method methodName="osresync.refreshOSCache" />
</methodCall>
```

Appendix A. WAAPI properties and command-line options

Use this information to learn about the properties and command-line options of the WAAPI client.

The properties file for the WAAPI client is located in *webgui-home*/waapi/etc/waapi.init. By default, when you run the **runwaapi** command, the properties specified in this file are used.

Tip: To specify a different properties file, use the -props option, for example:

```
runwaapi -props webgui-home/waapi/etc/test/waapi.init
```

Use the WAAPI properties file for property values that are not likely to change when the WAAPI client runs. For example, because the host name and port of the WAAPI client do not tend to change, set the following properties in the waapi.init file:

- **waapi.host**
- **waapi.port**

When you run the WAAPI client, to override a setting from the value in the waapi.init file, use the command-line option for that setting. For example, to specify a different XML command file for the WAAPI client to use, use the -file option:

```
runwaapi -file path_to_file
```

Where *path_to_file* overrides the value of the **waapi.file** property in the waapi.init file.

The default properties and corresponding command-line options for the WAAPI client are shown in the following table.

Table 42. WAAPI properties and command-line options

Property	Command-line option	Description
N/A	-help	Displays WAAPI command line help text.
N/A	-outfile *string*	The path of the output file.
N/A	-props *string*	Specifies a WAAPI properties file. The default is waapi.init.
Connection properties:		
waapi.host	-host *string*	The host name of the Web GUI server.
waapi.port	-port *integer*	The TCP/IP port on which the Web GUI server is running.
waapi.contextpath	N/A	The context that contains servlet's view of Web GUI within which the servlet is running.
waapi.secureport	-secureport *integer*	The TCP/IP port the Web GUI server listens on when in secure mode. The default port for HTTPS is 16316.

© Copyright IBM Corp. 2011, 2012

Table 42. WAAPI properties and command-line options (continued)

Property	Command-line option	Description
waapi.user	-user *string*	The name of the WAAPI user. This user must be assigned the ncw_admin Web GUI role but does not require any roles in the ObjectServer.
waapi.password	-password *string*	The user's password.
waapi.password.encryption	-password Encryption none | aes | fips	Indicates that passwords in the `waapi.init` file are encrypted. **AES** The password can be encrypted using the **ncw_aes_crypt** utility. **FIPS** The password can be encrypted using the **ncw_fips_encrypt** utility. If WAAPI is running in FIPS 140–2 mode, the `aes` option is not permitted.
waapi.file	-file *string*	The path and file name of the WAAPI configuration file.
waapi.timeoutsecs	**-timeout** *integer*	The time an inactive user's session remains active. After this time the network connection is released for use. The default time in seconds is 600.
Secure mode properties:		
waapi.secure	-secure off | on | fips	Enables the secure HTTPS features and FIPS 140–2 mode: **off** WAAPI supports HTTP only (no encryption) **on** WAAPI supports secure HTTPS **fips** WAAPI uses HTTPS in FIPS 140–2 mode encryption Can only be used if WAAPI runs with the JRE bundled with the Web GUI or an AIX® JRE The default is "`off`."
waapi.ssl.keyStore	-keyStore*string*	The location of the SSL key store.
waapi.ssl.keyStore Password	-keyStore Password*string*	The key store password.
waapi.ssl.key ManagerType	-keyManagerType *string*	The key manager type: • Set this property to `IbmX509` if you are using the JRE bundled with the Web GUI or an AIX JRE. • Set this property to `SunX509` if you are not using the bundled JRE or an AIX JRE. The default is `IbmX509`.

Table 42. WAAPI properties and command-line options (continued)

Property	Command-line option	Description
waapi.ssl.keyStore Type	-keyStoreType *string*	The key store type: • Set this property to `PKCS12` if you are using the JRE bundled with the Web GUI or an AIX JRE. • Set this property to `JKS` if you are not using the bundled JRE or an AIX JRE. The default is `PKCS12`.
waapi.ssl.trustStore	-trustStore *string*	The location of the SSL trust store.
waapi.ssl. trustStorePassword	-trustStore Password*string*	The trust store password.
waapi.ssl.trust ManagerType	-trustManager Type*string*	The key manager type: • Set this property to `IbmX509` if you are using the JRE bundled with the Web GUI or an AIX JRE. • Set this property to `SunX509` if you are not using the bundled JRE or an AIX JRE. The default is `IbmX509`.
waapi.ssl.trust StoreType	-trustStoreType *string*	The trust store type: • Set this property to `PKCS12` if you are using the JRE bundled with the Web GUI or an AIX JRE. • Set this property to `JKS` if you are not using the bundled JRE or an AIX JRE. The default is `PKCS12`.
waapi.fips.security. key	-fipsSecurityKey *string*	Points to the location where the security key is stored. The default is `%%/etc/encrypt/vault.key`.
waapi.ssl.protocol. handler.pkgs	-protocolHandler *string*	The SSL protocol handler package. The default is `com.ibm.net.ssl.www2.protocol`.

Table 42. WAAPI properties and command-line options (continued)

Property	Command-line option	Description
waapi.ssl.ip. authentication	-sslIPAuthen tication false | true	The default SSL host name verifier included in JVM 1.4 does not allow the validation of IP addresses as trusted host locations: **false** Optional: Use the default host name verifier to check connection credentials. **true** Use a custom host name verifier to compare the host name being connected to with the name of the peer in this SSL session. The host name of the peer does not have to be a fully qualified host name, or even a host name at all. It can represent a string encoding of the peer's network address. The host name is not authenticated. Enter a trusted IP address against which the host IP address is checked.
waapi.ssl.default. check	-defaultSslCheck false | true	If **waapi.ssl.ip.authentication** is set to `false`, this property specifies whether the host IP address that is connected is checked against the trusted IP address specified here. **false** No validation takes place **true** Validation takes place using the default host name verifier.
waapi.trusted.host.ip	-trustedHostIP *string, string*	If **waapi.ssl.ip.authentication** is set to true, this property specifies the list of trusted IP addresses that the host IP address that is connected to is checked against. **Note:** Multiple IP addresses are specified in a comma-separated list.
Log file properties:		
waapi.logfile	**-logfile***string*	The location of WAAPI log files and directories

Appendix B. Installing the WAAPI client on a remote host

Additionally to the WAAPI client on the Web GUI server, you can install the WAAPI client on a remote host.

About this task

To install the WAAPI client on a remote host:

Procedure

1. Log in to the Web GUI using any suitable user ID.
2. On the navigation pane, click **Welcome** and then click **WAAPI Client Information** in the work area.
3. Download the WAAPI client:
   - UNIX Linux Click **waapi.tar.gz**
   - Windows Click **waapi.zip**

   Save the file to the directory where you want to install the client.
4. Change to the directory in which you want the files to be installed.
5. Uncompress the file.

   Tip: UNIX Use the **gtar xzvf waapi.tar.gz** command or the **gunzip waapi.tar.gz** and **tar -xvf waapi.tar** commands.
   The files are installed in a subdirectory called `waapi` with the following structure:
   - `waapi/`
   - `waapi/bin`
   - `waapi/etc/`
   - `waapi/etc/default`
   - `waapi/etc/docs`
   - `waapi/etc/samples/`
   - `waapi/log/`
   - `waapi/platform/java/bin`
   - `waapi/platform/java/lib`
6. UNIX Set the execute-permissions on the **runwaapi** command:

   ```
   chmod +x waapi/bin/runwaapi
   ```

Results

You can now use the WAAPI client on the remote host.

© Copyright IBM Corp. 2011, 2012

Appendix C. WAAPI security

WAAPI has a number of security features that help to ensure secure communication with the Web GUI server.

Administration of any system relies upon security. WAAPI provides a number of security features that you can use:

- A secure connection to the Web GUI server.

  Consider using an SSL connection, with or without FIPS-142 protection, when running the WAAPI client on a server remote from the Web GUI server.
- Encrypting WAAPI passwords.

  You can encrypt WAAPI passwords using either AES or FIPS-142 (this is required when the connection ise secured using FIPS-142). You can use this feature whether the WAAPI client is installed with or remote from the Web GUI server.
- Securing the `waapi.init` file

  The properties file for the WAAPI client contains a number of sensitive entries. For example, there is the username and password of the user to use on the Web GUI server to run WAAPI commands. So it is important to ensure that the file is available only to authorized administrators.

WAAPI provides the following security tasks:

- "Creating secure WAAPI connections"
- "Enabling WAAPI password encryption" on page 106
- "Securing the waapi.init properties file" on page 108

Creating secure WAAPI connections

You can configure the Web GUI to use the Secure Sockets Layer (SSL) protocol for secure WAAPI communication, using either server-only authentication, or server-and-client authentication. Optionally, you can enable the FIPS 140-2 mode.

About this task

The following WAAPI connection modes can be configured for Web GUI:

SSL not enabled
: The WAAPI client connects to the Web GUI via standard HTTP. This mode does not require any additional configuration.

SSL enabled, server-only authentication (without FIPS 140–2)
: The WAAPI client connects to the Web GUI via HTTPS using server-only authentication, but not in FIPS 140–2 mode.

SSL enabled, server-and-client authentication (without FIPS 140–2)
: The WAAPI client connects to the Web GUI via HTTPS using server-and-client authentication, but not in FIPS 140–2 mode.

SSL enabled, server-only authentication with FIPS 140–2
: The WAAPI client connects to the Web GUI via HTTPS using server-only authentication in FIPS 140–2 mode.

© Copyright IBM Corp. 2011, 2012

SSL enabled, server-and-client authentication with FIPS 140–2
The WAAPI client connects to the Web GUI via HTTPS using server-and-client authentication in FIPS 140–2 mode.

Creating a WAAPI SSL connection (server-only authentication)

To create a secure, server-authenticated connection between WAAPI and the Web GUI deployed within Tivoli Integrated Portal (without FIPS 140–2), you must reference Tivoli Integrated Portal in the WAAPI truststore as well as the `waapi.init` file.

About this task

To create a secure, server-authenticated connection between WAAPI and the Web GUI:

Procedure

1. Using the Tivoli Integrated Portal GUI, extract the default truststore signer certificate.
   a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
   b. Click **Security** > **SSL certificate and key management** > **Key stores and certificates** > **NodeDefaultKeyStore** > **Personal certificates**.
   c. Select the default (Alias) truststore certificate and click **Extract**.
   d. Type a name, for example, `/example/tipcert.arm`.
   e. Select **Base64-encoded ASCII data** and click **Ok**.
2. Using the Tivoli Integrated Portal Ikeyman utility, add the new certificate to the WAAPI truststore.
   a. Go to *tip_home_dir*`/bin` and start Ikeyman.
   b. Click **KeyDatabaseFile** > **New** and select **PKCS** as the key database type.
   c. Type a truststore name, for example `/example/waapiTruststore.p12`.
   d. Enter the default password `WebAS` and click **Ok**.
   e. Select **Signer Certificates** from the dropdown list and click **Add**.
   f. Point to the signer certificate, in this example */example/tipcert.arm*, and click **Ok**. Make a note of the signer certificate CN (common name) value.
3. Edit the `waapi.init` file.
   a. Open *webgui-home*`/waapi/etc/waapi.init` and go to the WAAPI Secure Modes section.
   b. Set **waapi.secure:on**.
   c. Ensure that the host name in **`waapi.host`** is the same as the CN (common name) value in the signer certificate.
   d. Provide the truststore name, in this example */example/waapiTruststore.p12*.
   e. Enter the password `WebAS`.

What to do next

To test if you have successfully set up the WAAPI SSL connection, execute a WAAPI example.

Creating a WAAPI SSL connection (client-server authentication)

To create a secure, client- and server-authenticated connection between WAAPI and the Web GUI deployed within Tivoli Integrated Portal (without FIPS 140–2), you reference Tivoli Integrated Portal in the WAAPI truststore and WAAPI in the Tivoli Integrated Portal truststore. You then enable SSL authentication in WAAPI and add the WAAPI keystore certificate to your browser's truststore. Lastly, you enable client authentication in Tivoli Integrated Portal.

Procedure

1. Using the Tivoli Integrated Portal GUI, extract the default truststore signer certificate.
   a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
   b. Click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates**.
   c. Select the default (Alias) truststore certificate and click **Extract**.
   d. Type a name, for example, */example/tipcert.arm*.
   e. Select **Base64-encoded ASCII data** and click **Ok**.
2. Using the Tivoli Integrated Portal Ikeyman utility, add the new certificate to the WAAPI truststore.
   a. Go to *tip_home_dir*/bin and start Ikeyman.
   b. Click **KeyDatabaseFile > New** and select **PKCS** as the key database type.
   c. Provide a truststore name, for example */example/waapiTruststore.p12*.
   d. Enter the default password WebAS and click **Ok**.
   e. Select **Signer Certificates** from the dropdown list and click **Add**.
   f. Point to the signer certificate, in this example */example/tipcert.arm*, and click **Ok**. Make a note of the signer certificate CN (common name) value.
3. Using the Tivoli Integrated Portal Ikeyman utility, extract a self-signed personal keystore certificate from the WAAPI keystore.
   a. Go to *tip_home_dir*/bin and start Ikeyman.
   b. Click **KeyDatabaseFile > New** and select **PKCS** as the key database type.
   c. Provide a keystore name, for example *waapiKeystore.p12*.
   d. Enter the default password WebAS and click **Ok**.
   e. Select **Personal Certificates** from the dropdown list and click **New Self-Signed**.
   f. Enter a key label, for example WAAPI_cert, complete the other fields as required, then click **Ok**.
   g. Select the new keystore certificate, in this example WAAPI_cert, and click **Extract Certificate**.
   h. Select **Base64-encoded ASCII data**.
   i. Enter a certificate file name, for example WAAPI_cert.arm, and define a location, in this example /example/, then click **Ok**.
4. Using the Tivoli Integrated Portal GUI, add the new WAAPI keystore certificate to the Tivoli Integrated Portal truststore.
   a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
   b. Click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates**.

c. Click **Add** and enter an alias of `WAAPI_cert` (for this example).
d. Point to the previously-generated WAAPI_cert, click **Ok**, then **Save**.

5. Using your browser's security management functionality, add the new keystore certificate to the browser's truststore.
   Warning: If you do not complete this step, you will no longer be able to access Tivoli Integrated Portal after you enable client authentication in the next step.
6. Using the Tivoli Integrated Portal GUI, enable client authentication.
   a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
   b. Click **Security > SSL certificate and key management > SSL Configurations > NodeDefaultSSLSettings > Quality of protection (QoP) settings**.
   c. Select **Required** from the **General Properties > Client authentication** drop-down list.
   d. Click **Ok**, then **Save**.
7. Edit the `waapi.init` file.
   a. Open *webgui-home*`/waapi/etc/waapi.init` and go to the WAAPI Secure Modes section.
   b. Set **waapi.secure:on**.
   c. Ensure that the host name in waapi.host is the same as the CN (common name) value in the signer certificate.
   d. Provide the keystore name, in this example */example/waapiKeystore.p12*.
   e. Provide the truststore name, in this example */example/waapiTruststore.p12*.
   f. Enter the password of `WebAS`.

   Note: Windows When entering the location of keystore and truststore on a Windows system, use two backslashes as the path separator because a single backslash is interpreted as an escape character. For example to specify the truststore use *example**waapiTruststore.p12*.

What to do next

To test if you have successfully set up the WAAPI SSL connection, execute a WAAPI example.

Creating a WAAPI SSL connection with FIPS 140–2 (server-only authentication)

To create a secure, server-authenticated connection between WAAPI and the Web GUI deployed within Tivoli Integrated Portal with FIPS 140–2 mode enabled, you must reference Tivoli Integrated Portal in the WAAPI truststore as well as the waapi.init file. You must then enable FIPS 140–2 in Tivoli Integrated Portal.

About this task

If you have already enabled FIPS 140–2 in Tivoli Integrated Portal while setting up FIPS 140–2 for the Web GUI, you do not need to complete step four.

Procedure

1. Using the Tivoli Integrated Portal GUI, extract the default truststore signer certificate.

a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
b. Click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates**.
c. Select the default (Alias) truststore certificate and click **Extract**.
d. Give it a name, for example, */example/tipcert.arm*.
e. Select **Base64-encoded ASCII data** and click **Ok**.

2. Using Tivoli Integrated Portal's Ikeyman utility, add the new certificate to the WAAPI truststore.
   a. Go to *tip_home_dir*/bin and start Ikeyman.
   b. Click **KeyDatabaseFile > New** and select **PKCS** as the key database type.
   c. Provide a truststore name, for example */example/waapiTruststore.p12*.
   d. Enter the default password WebAS and click **Ok**.
   e. Select **Signer Certificates** from the dropdown list and click **Add**.
   f. Point to the signer certificate, in this example */example/tipcert.arm*, and click **Ok**. Make a note of the signer certificate CN (common name) value.
3. Edit the waapi.init file.
   a. Open *webgui-home*/waapi/etc/waapi.init and go to the WAAPI Secure Modes section.
   b. Set **waapi.secure:fips**.
   c. Ensure that the host name in waapi.host is the same as the CN (common name) value in the signer certificate.
   d. Provide the truststore name, in this example */example/waapiTruststore.p12*.
   e. Enter the password of WebAS.
4. Enable FIPS 140–2 in the Tivoli Integrated Portal server.
   a. Open *webgui-home*/java/jre/lib/security/java.security.
   b. In the list of providers and their order of preference, uncomment security.provider:<*x*>=com.ibm.crypto.fips.provider.IBMJCEFIPS
   c. Replace the <*x*> variable with 1 and re-number the subsequent security providers.
   d. Using the Tivoli Integrated Portal GUI, enable FIPS 140–2.
   Click **Security > SSL Certificate and key management** and select the **FIPS** checkbox under Configuration Settings, then click **Apply**.
   e. Restart the Tivoli Integrated Portal server.

Example

What to do next

To test if you have successfully set up the WAAPI SSL connection, execute a WAAPI example.

Creating a WAAPI SSL connection with FIPS 140–2 (client-server authentication)

To create a secure, client- and server-authenticated connection between WAAPI and the Web GUI deployed within Tivoli Integrated Portal with FIPS 140–2 mode enabled, you reference Tivoli Integrated Portal in the WAAPI truststore and WAAPI in the Tivoli Integrated Portal truststore. You then enable FIPS 140–2 authentication in WAAPI and add the WAAPI keystore certificate to your browser's truststore. Lastly, you enable client authentication and FIPS 140–2 in Tivoli Integrated Portal.

About this task

If you have already enabled FIPS 140–2 in Tivoli Integrated Portal while setting up FIPS 140–2 for the Web GUI, you do not need to complete step eight.

Procedure

1. Using the Tivoli Integrated Portal GUI, extract the default truststore signer certificate.
   a. Click **Settings** > **WebSphere Admin Console**, and click **Launch WebSphere Admin Console**.
   b. Click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates**.
   c. Select the default (Alias) truststore certificate and click **Extract**.
   d. Give it a name, for example, */example/tipcert.arm*.
   e. Select **Base64-encoded ASCII data** and click **Ok**.
2. Using the Tivoli Integrated Portal Ikeyman utility, add the new certificate to the WAAPI truststore.
   a. Go to *tip_home_dir*/bin and start Ikeyman.
   b. Click **KeyDatabaseFile > New** and select **PKCS** as the key database type.
   c. Provide a truststore name, for example */example/waapiTruststore.p12*.
   d. Enter the default password WebAS and click **Ok**.
   e. Select **Signer Certificates** from the dropdown list and click **Add**.
   f. Point to the signer certificate, in this example */example/tipcert.arm*, and click **Ok**. Make a note of the signer certificate CN (common name) value.
3. Using Tivoli Integrated Portal's Ikeyman utility, extract a self-signed personal keystore certificate from the WAAPI keystore.
   a. Go to *tip_home_dir*/bin and start Ikeyman.
   b. Click **KeyDatabaseFile > New** and select **PKCS** as the key database type.
   c. Provide a keystore name, for example *waapiKeystore.p12*.
   d. Enter the default password WebAS and click **Ok**.
   e. Select **Personal Certificates** from the dropdown list and click **New Self-Signed**.
   f. Enter a key label, for example WAAPI_cert, complete the other fields as required, then click **Ok**.
   g. Select the new keystore certificate, in this example WAAPI_cert, and click **Extract Certificate**.
   h. Select **Base64-encoded ASCII data**.
   i. Enter a certificate file name, for example WAAPI_cert.arm, and define a location, in this example /example/, then click **Ok**.

4. Using the Tivoli Integrated Portal GUI, add the new keystore certificate to the Tivoli Integrated Portal truststore.
   a. Click **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates**.
   b. Click **Add** and enter an alias of `WAAPI_cert` (for this example).
   c. Point to the previously generated WAAPI_cert, click **Ok**, then **Save**.
5. Using your browser's security management functionality, add the new keystore certificate to the browser's truststore.
   Warning: If you do not complete this step, you will no longer be able to access Tivoli Integrated Portal after you enable client authentication in the next step.
6. Using the Tivoli Integrated Portal GUI, enable client authentication.
   a. Click **Security > SSL certificate and key management > SSL Configurations > NodeDefaultSSLSettings > Quality of protection (QoP) settings**.
   b. Select **Required** from the **General Properties > Client authentication** drop-down list.
   c. Click **Ok**, then **Save**.
7. Edit the `waapi.init` file.
   a. Open *webgui-home*`/waapi/etc/waapi.init` and go to the WAAPI Secure Modes section.
   b. Set **waapi.secure:fips**.
   c. Ensure that the host name in **`waapi.host`** is the same as the CN (common name) value in the signer certificate.
   d. Provide the truststore name, in this example */example/waapiTruststore.p12*.
   e. Enter the password of `WebAS`.
8. Enable FIPS 140–2 in the Tivoli Integrated Portal server.
   a. Open *tip_home_dir*`/java/jre/lib/security/java.security`.
   b. In the list of providers and their order of preference, uncomment `security.provider:<x>=com.ibm.crypto.fips.provider.IBMJCEFIPS`
   c. Replace the <*x*> variable with 1 and re-number the subsequent security providers.
   d. Using the Tivoli Integrated Portal GUI, enable FIPS 140–2.
      Click **Security > SSL Certificate and key management** and select the **FIPS** checkbox under Configuration Settings, then click **Apply**.
   e. Restart the Tivoli Integrated Portal server.

What to do next

To test if you have successfully set up the WAAPI SSL connection, execute a WAAPI example.

Enabling WAAPI password encryption

You can opt to store WAAPI passwords in the `waapi.init` file in encrypted format.

About this task

For password encryption for non-SSL and SSL connections, you use AES encryption, while for FIPS 140-2 connections, you use FIPS 140–2 mode encryption. The encryption types, and the tools required to enable them, are as follows:

Non-SSL (HTTP) connections
Passwords can be AES encrypted using the **ncw_aes_crypt** tool.

SSL (HTTPS) connections
Passwords can be AES encrypted using the **ncw_aes_crypt** tool.

SSL (HTTPS) connections with FIPS 140–2 enabled
Passwords can be encrypted using the **ncw_fips_crypt** tool.

WAAPI passwords must be encrypted using the **ncw_fips_crypt** script in *webgui-home*/waapi/bin. This script uses the vault key in *waapi_install_dir*/etc/encrypt. If it does not already exist, the vault.key file is automatically generated on the first execution of the script.

Encrypting WAAPI passwords using AES

To encrypt WAAPI passwords for non-SSL and SSL connections, use the **ncw_aes_crypt** tool. You can configure WAAPI password encryption for WAAPI clients installed on the same server as Tivoli Integrated Portal or on a different server.

Before you begin

You can use AES password encryption only if FIPS 140–2 mode has not been enabled.

About this task

The default truststore password is WebAS.

The location of the `waap.init` file, in which you must enter the encrypted password, differs depending on where the WAAPI client is installed. If the WAAPI client is installed on the same server as Tivoli Integrated Portal, the file is located at *webgui-home*/waapi/etc/waapi.init. If the WAAPI client is installed on a different server than Tivoli Integrated Portal, the file is located at *waapi_install_dir*/etc/waapi.init.

Procedure

1. Encrypt the WAAPI password:
   a. Run *webgui-home*/waapi/bin/ncw_aes_crypt.
   b. Enter the default Tivoli Integrated Portal truststore password WebAS. An encrypted password is generated.
   c. Copy the encrypted password.
2. Add the encrypted password:
   a. Open the waapi.init file.
   b. Set the **waapi.password.encryption** property to aes.

c. Set the **waapi.ssl.trustStorePassword** property to the password encrypted in step 1 on page 106.

3. Repeat steps 2b on page 106 and 2c for the following properties:
   - **waapi.password**
   - **waapi.ssl.keyStorePassword**

Encrypting WAAPI passwords using FIPS 140–2 mode encryption

To encrypt WAAPI passwords for non-SSL and SSL connections in FIPS 140–2 mode, use the **ncw_fips_crypt** tool. You can configure WAAPI password encryption whether the WAAPI client is installed on the same server as Tivoli Integrated Portal or on a different server.

Before you begin

If FIPS 140–2 mode has been enabled, you can use only FIPS 140–2 mode password encryption. You must have IBM® JRE installed to use the **ncw_fips_crypt** tool.

About this task

The default truststore password is WebAS.

The default WAAPI vault (secret) key used is located in *webgui-home*/waapi/etc/encrypt/vault.key. The vault key is automatically be generated on first use of the **ncw_fips_crypt** tool and stored in the *waapi_install_dir*/etc/encrypt/vault.key file.

The location of the waap.init file, in which you must enter the encrypted password, differs depending on where the WAAPI client is installed. If the WAAPI client is installed on the same server as Tivoli Integrated Portal, the file is located at *webgui-home*/waapi/etc/waapi.init. If the WAAPI client is installed on a different server than Tivoli Integrated Portal, the file is located at *waapi_install_dir*/etc/waapi.init.

Procedure

1. Encrypt the WAAPI password:
   a. Enter the following command:
   webgui-home/waapi/bin/ncw_fips_crypt –password *WebAS* -key *webgui-home*/waapi/etc/encrypt/vault.key
   If you use the default vault key, omit the **key** parameter. An encrypted password is generated
   b. Copy the encrypted password.
2. Add the encrypted password:
   a. Open the waapi.init file.
   b. Set the **waapi.password.encryption** property to fips
   c. Set the **waapi.ssl.trustStorePassword** property to the password generated in step 1.
3. Repeat steps 2b and 2c for the following properties:
   - **waapi.password**
   - **waapi.ssl.keyStorePassword**

What to do next

To generate a new vault key, use the **-genkey** parameter. Enter the following command: *webgui-home*/waapi/bin/ncw_fips_crypt –genkey <*locationofvaultkeyfile*>. After the command has run, copy the new vault key file to the *waapi_install_dir*/etc/encrypt/ directory.

Securing the waapi.init properties file

Protect the waapi.init properties file from access by unauthorized users.

Procedure

Use the facilities of the operating system to set the access permissions to the waapi.init file. Ensure that the file is accessible only to authorized administrators of the Web GUI server. This is especially important when the file contains the username and password of the Web GUI administrator.

Notices

This information was developed for products and services offered in the U.S.A.

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
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 2011, 2012

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
958/NH04
IBM Centre, St Leonards
601 Pacific Hwy
St Leonards, NSW, 2069
Australia

IBM Corporation
896471/H128B
76 Upper Ground
London SE1 9PZ
United Kingdom

IBM Corporation
JBFA/SOM1
294 Route 100
Somers, NY, 10589-0100
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Portions of this product include software developed by Daniel Veillard.

- `libxml2-2.7.8`

The `libxml2-2.7.8` software is distributed according to the following license agreement:
© Copyright 1998-2003 Daniel Veillard.
All Rights Reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com®, Netcool, Netcool/OMNIbus, Tivoli®, and WebSphere® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both

Adobe, Acrobat, Portable Document Format (PDF), PostScript, and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.



Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

<filterCollection> 75
<icon> 36
<menu> 51
<metric:metric> 79
<relationship:relationship> 86
<resource> 46
<resources> 46
<separator> 51
<supermenu> 51
<tool:access> 55
<tool:cgiurl> 55
<tool:cmdline> 55
<tool:criterion> 55
<tool:equals> 55
<tool:fieldlist> 55
<tool:journal> 55
<tool:osfield> 55
<tool:script> 55
<tool:security> 55
<tool:sql> 55
<tool:tool> 55
<user> 12

A

accessibility viii
audience v
authentication
  client-server 101
    FIPS 140-2 104
  server-only 100
    FIPS 140-2 102

C

CGI request 66
  create or replace 67
  modify 67
  register 66
  unregister 67
client-server
  security 101
    FIPS 140-2 104
commands
  runwaapi
    command-line options 93
communications 3
comparison
  WAAPI and Web GUI procedure 2
conventions, typeface viii

D

Document Type Definition 9
DTD
  See Document Type Definition

E

education
  see Tivoli technical training viii
element
  <filterCollection> 75
  <icon> 36
  <menu> 51
  <metric:metric> 79
  <relationship:relationship> 86
  <resource> 46
  <resources> 46
  <separator> 51
  <supermenu> 51
  <tool:access> 55
  <tool:cgiurl> 55
  <tool:cmdline> 55
  <tool:criterion> 55
  <tool:equals> 55
  <tool:fieldlist> 55
  <tool:journal> 55
  <tool:osfield> 55
  <tool:script> 55
  <tool:security> 55
  <tool:sql> 55
  <tool:tool> 55
  <user> 12
environment variables, notation viii

F

file request 48
  add a directory 48
  add a file 49
  create or replace a file 50
  delete a file 50
  recursively remove a directory 51
  remove a directory 50
files
  waapi.init 93, 100, 101, 102, 104, 106, 107, 108
filter collection request 75
  add filter 76
  create 75
  create or replace 76
  delete 77
  delete filter 77
  list 77
  modify 78
  set view 78
filter request 68
  add 68
  create or replace 72
  delete 73
  list 73
  modify 73
  set default view 74
FIPS 140-2
  client-server 104
  server-only 102

L

load balancing
  remove node 90, 92

M

manuals vi
map request 26
  create 27
  create or replace 43
  delete 43
  list 44
  modify 44
map visual request
  add 44
  add or replace 45
  delete 45
  modify 46
menu request 51
  create 51
  create or replace 53
  delete 54
  list 54
  modify 54
metric request 79
  create 79, 86
  create or replace 83
  delete 84
  list 84
  modify 85

O

online publications vi
ordering publications vi
other request 90
  reload filters and views 91, 92
  remove a node from a cluster 90, 92
  resynchronize the cache 90, 92
  system status report 91

P

password
  encryption
    AES 106
    FIPS 140–2 107
passwords
  encryption 4, 106
prompt request 64
  create or replace 64
  delete 65
  list 65
  modify 65
publications vi

© Copyright IBM Corp. 2011, 2012

R

relationship request 86
  create or replace 88
  delete 89
  list 89
  modify 89
remote host
  installation
    WAAPI client 97
request 7
  CGI 66
  file 48
  filter 68
  filter collection 75
  map 26
  menu 51
  metric 79
  other 90
  prompt 64
  relationship 86
  resource 46
  structure 7
  tool 55
  user 11
  view 19
resource request 46
  add 46
  create or replace 47
  list 48
  remove 48

S

security 99
  FIPS 140-2
    client-server 104
    server-side 102
  password encryption 4, 106
    AES 106
    FIPS 140-2 107
  WAAPI
    client-server 101
    overview 4, 99
    server-side 100
server
  security 100
    FIPS 140-2 102
SSL
  password encryption
    AES 106
support information viii

T

Tivoli software information center vi
Tivoli technical training viii
tool request 55
  create 55
  create or replace 62
  delete 62
  list 63
  modify 63
training, Tivoli technical viii
truststore
  WAAPI
    signer certificate 100, 101, 102, 104
typeface conventions viii

U

user request 11
  list 19
  maintain 11
  modify 11
utilities
  ncw_aes_crypt 106
  ncw_fips_crypt 107

V

variables, notation for viii
view request 19
  create 19
  create or replace 24
  delete 26
  list 26
  modify 25

W

WAAPI
  command-line options 93
  communication with the server 3
  comparison with Web GUI 2
  installation
    remote host 97
  password encryption 4, 106
    AES 106
    FIPS 140–2 107
  properties 93, 108
  security 99
    overview 4, 99
  truststore
    signer certificate 100, 101, 102, 104
  using 5
WAAPI request 7
  case senstitivity 9
  CGI 66
  characteristics 9
  comments 9
  content and value restrictions 9
  Document Type Definition 9
  file 48
  filter 68
  filter collection 75
  map 26
  menu 51
  metric 79
  order of elements 9
  other 90
  prompt 64
  relationship 86
  resource 46
  root element 7
  structure 7
  tool 55
  user 11
WAAPI request *(continued)*
  view 19
  XML declaration 7



Printed in the Republic of Ireland

SC14-7535-00

