Netcool Agile Service Manager
Version 1.1.5

Installation, Administration and User Guide

08 July 2019

Netcool Agile Service Manager
Version 1.1.5

Installation, Administration and User Guide

08 July 2019

Note
Before using this information and the product it supports, read the information in "Notices" on page 319.

This edition applies to Version 1.1.4 of IBM Netcool Agile Service Manager (product number 5725-Q09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Tables

Preface

This PDF document contains topics from the Knowledge Center in a printable format.

About the latest release

Agile Service Manager Version 1.1.5 is available.

What's new in Version 1.1.5

Software released: 28 June 2019

Documentation updated: 08 July 2019

ICP on OpenShift

New capability to deploy Agile Service Manager on IBM Cloud Private with Red Hat OpenShift:
https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.2.0/supported_environments/openshift/overview.html

New 'ICP on OpenShift' reference topic describing configuration requirements specific to Agile Service Manager:
"ICP on OpenShift reference" on page 43

New process to define alternative storage for ICP on OpenShift:
"Configuring the Helm chart to use alternate storage (ICP on OpenShift)" on page 207

Requirements

Updated hardware and software requirements:
Chapter 2, "Planning," on page 9

Details of sizing for basic (size0) and production (size1) systems:
"Sizing reference" on page 313

Installation

New process to install a hybrid system:
"Installing and configuring a hybrid ICP / on-prem system" on page 46

List of configurable installation parameters:
"Installation parameters" on page 316

Authentication between services

New authentication section:
"Configuring core services authentication" on page 183

User administration

Refined user roles:
"Configuring DASH user roles" on page 21

Backup and restore

New database backup for ICP:
"Backing up database data (ICP)" on page 213
"Restoring database data (ICP)" on page 216

Topology rules

Refined ability to create rules for resource data:
"Defining rules" on page 223

Topology viewer

New status tools:
"Configuring custom tools" on page 192

New resource annotation feature, as well as more granularity in exploring a resource's connections (resource neighbors and relationships):
"Viewing a topology" on page 168

New filtering:
"Rendering (visualizing) a topology" on page 166

Enhanced resource type styling:
"Editing resource type styles" on page 199

Refined direct-launch URL strings:
"Accessing topologies via direct-launch URL string" on page 164

New observers

AWS Observer:
"Configuring AWS Observer jobs" on page 56
"Defining AWS Observer jobs" on page 102

Zabbix Observer:
"Configuring Zabbix Observer jobs" on page 98
"Defining Zabbix Observer jobs" on page 160

Ciena Blue Planet Observer:
"Configuring Ciena Blue Planet Observer jobs" on page 59
"Defining Ciena Blue Planet Observer jobs" on page 106

Chapter 1. Product overview

IBM Netcool Agile Service Manager provides operations teams with complete up-to-date visibility and control over dynamic infrastructure and services. Agile Service Manager lets you query a specific networked resource, and then presents a configurable topology view of it within its ecosystem of relationships and states, both in real time and within a definable time window. **Agile Service Manager is available as both on-prem and IBM Cloud Private versions.**

Benefits of Netcool Agile Service Manager

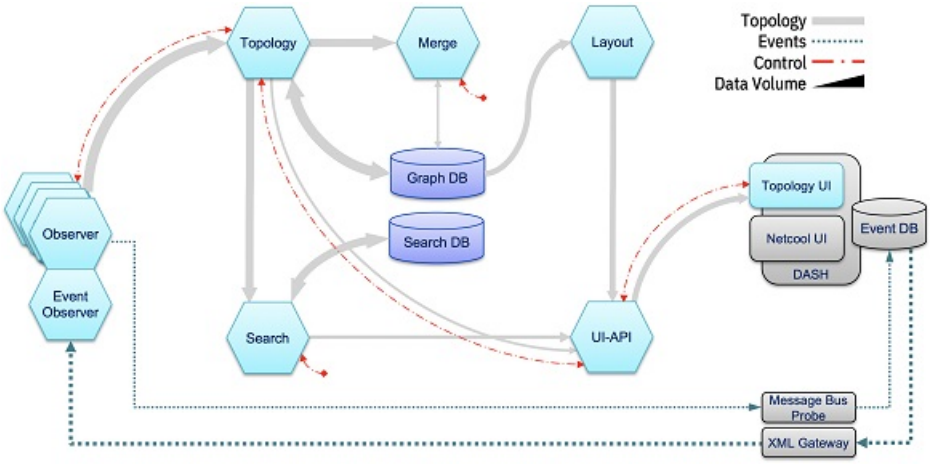
Services and applications are increasingly deployed in environments that take advantage of distributed and often virtualized infrastructure. For example, parts of a network might be cloud-based, with other connected elements contained within, or tethered to, legacy systems that exploit tried and tested on-prem capability. The result is often a highly distributed and increasingly complex hybrid network that requires an agile and dynamic operations management solution in order to leverage and exploit its rapidly evolving technologies.

Netcool Agile Service Manager allows the real-time view, support and management of highly dynamic infrastructures and services. By visualizing complex network topologies in real-time, updated dynamically or on-demand, and allowing further investigation of events, incidents and performance, operational efficiency is improved, problems are detected and solved faster, false alarms are reduced, and automation and collaboration between operational teams is improved. Also, data can be leveraged more efficiently both in real time and historically, thereby empowering teams and systems to create and nurture differentiated services for different customers.

IBM Netcool Agile Service Manager is cloud-born, and built on secure, robust and proven technologies. It is designed to be flexible and can be extended as needed using plug-in components and micro-services to cater for highly specific environments.

Basic deployment

Netcool Agile Service Manager is deployed with IBM Tivoli Netcool Operations Insight as part of an integrated solution. This figure depicts the basic Agile Service Manager **on-prem** architecture.



Deployment scenarios

Network Manager

You want to use Netcool Agile Service Manager to analyze the resource data discovered by Network Manager.

You configure the ITNM Observer to load topology data, and then monitor Network Manager for updates.

You define a seed resource in the Agile Service Manager UI, and then dynamically render a topology view centered around that resource, which can display linked resources up to four hops away.

You use this visualization to delve into the states, histories and relationships of the resources displayed.

New data is harvested continuously, which you can then analyze further.

Netcool/OMNIbus

You want to extend your analysis of Netcool/OMNIbus events.

You configure the Event Observer and the Netcool/OMNIbus XML Gateway and Message Bus to monitor the Netcool/OMNIbus ObjectServer for new events.

You configure the IBM Tivoli Netcool/OMNIbus Probe for Message Bus to synchronize event views across the Netcool Agile Service Topology Viewer and the Netcool/OMNIbus Event Viewer.

You display a topology based on a specific resource (event), and then exploit Netcool Agile Service Manager's functionality to gain further insights into the displayed events.

OpenStack

You use the OpenStack Observer to render detailed OpenStack topologies, and delve further into their states, histories and relationships.

Bespoke topologies using the REST APIs

You want to load resource data from your own source in order to use the Netcool Agile Service Manager functionality to render topologies for analysis.

You use the REST APIs to configure a data source, load your data, and then use the Netcool Agile Service Manager UI to focus on a specific seed resource, before extending your topology outward.

Components

Netcool Agile Service Manager consists of a number of services, and can be integrated into the IBM Netcool Operations Insight suite of products. You access Netcool Agile Service Manager through the IBM Dashboard Application Service Hub (DASH).

Agile Service Manager core download packages

The Agile Service Manager core eAssembly consists of the following packages. Apart from the UI, which is installed using the IBM Installation Manager, all core packages are Docker containers.

Table 1. Agile Service Manager core packages

Package	Details
com.ibm.itsm.topology.ui.zip	The Agile Service Manager user interface, which presents you with a topology view and lets you perform a number of further tasks in context. Once installed, this interface is accessed through DASH.
nasm-cassandra	A distributed and robust database that is scalable while maintaining high performance.
nasm-common	Contains the product licenses, common scripts and docker-compose.
nasm-elasticsearch	A distributed search and analytics engine that is scalable and reliable.
nasm-kafka	A message bus that efficiently consolidates topology data from multiple sources. In addition to the Kafka message bus, the nasm-kafka service also deploys the Kafka REST API, which verifies the existence of Kafka topics.
nasm-layout	A service that lets you customize the way topologies are structured, providing a number of standard options, such as hierarchical, force-directed, and other views.
nasm-merge	A service that lets you merge duplicate records of the same resource retrieved through different mechanisms into one composite resource.
nasm-proxy	A service that manages access to all other Agile Service Manager micro-services.
nasm-search	A service that inserts topology data into the Elasticsearch engine, and exposes REST APIs to search for resources.
nasm-topology	The service that lets you query networked resources, and retrieve both real-time and historical information about their state and relationships with other linked resources.
nasm-ui-api	A service whose dedicated purpose is to provide topology-related data to the Agile Service Manager UI.
nasm-zookeeper	A robust, distributed and scalable synchronization service.

Agile Service Manager observer download packages

Table 2. Agile Service Manager observer packages

Package	Details
nasm-alm-observer	A service that extracts information from the IBM Agile Lifecycle Manager.
nasm-aws-observer	A service that reads data from the Amazon Web Services
nasm-bigfixinventory-observer	A service that reads data from a Bigfix Inventory instance through its REST API
nasm-cienablueplanet-observer	A service that retrieves topology data from the Blue Planet MCP instance via REST API.
nasm-ciscoaci-observer	A service that makes REST calls to Cisco APIC in the Cisco ACI environment.
nasm-contrail-observer	A service that makes REST calls to the Contrail API server to retrieve topology data from Juniper Network Contrail.
nasm-dns-observer	A service that queries internal DNS servers, and returns response times and service addresses.
nasm-docker-observer	A service that extracts information from Docker networks.
nasm-dynatrace-observer	A service that queries a specified Dynatrace environment for information about its applications, services, process groups, and infrastructure entities
nasm-event-observer	A service that extracts information from IBM Tivoli Netcool/OMNIbus events.
nasm-file-observer	A service that retrieves data written to a file in a specific format.
nasm-ibmcloud-observer	A service that performs REST calls to the IBM Cloud REST API, which retrieve Cloud Foundry Apps information and services.
nasm-itnm-observer	A service that extracts information from the IBM Tivoli Network Manager IP Edition database.
nasm-kubernetes-observer	A service that discovers Kubernetes services containers and maps relationships between them.
nasm-newrelic-observer	A service that loads New Relic Infrastructure resource data via a New Relic account with a New Relic Infrastructure subscription.
nasm-openstack-observer	A service that extracts information from OpenStack.
nasm-rest-observer	A service that obtains topology data via REST endpoints.
nasm-servicenow-observer	A service that performs REST calls to retrieve configuration management database (CMDB) data from ServiceNow.
nasm-taddm-observer	A service that extracts information from the IBM Tivoli Application Dependency Discovery Manager database.
nasm-vmvcenter-observer	A service that dynamically loads VMware vCenter data.
nasm-vmwarensx-observer	A service that dynamically loads VMware NSX data.
nasm-zabbix-observer	A service that extracts server information and its associated network resources from Zabbix via REST RPC.

Related reference:

"Swagger reference" on page 311
Specific links to Agile Service Manager Swagger documentation are included in many of the topics, as and when useful. This topic summarizes some of that information in a single location, for example by listing the default ports and Swagger URLs for each Agile Service Manager service.

Related information:

IBM Netcool Agile Service Manager download document

Glossary

Refer to the following list of terms and definitions to learn about important Netcool Agile Service Manager concepts.

Netcool Agile Service Manager terminology

edge An edge is a relationship between resources, also simply referred to as the 'link' between resources.

Edges have a *label*, which allocates them to a family of edges with specific behavior and governs how they are displayed in the UI, and an *edgeType*, which defines the relationship in real terms.

hop A hop is a step along a single edge from one resource to another.

All resources that are connected directly to a seed resource are one hop removed, while those connected to the secondary resources are two hops removed from the seed resource, and so on.

Netcool Agile Service Manager displays topologies with resources up to four hops removed from the seed resource by default, which is configurable up to 30 hops.

CAUTION: Do not increase the hop count beyond your system's ability to cope. A large hop count can result in a very large topology, and rendering this can lead to timeout errors.

ICP (IBM Cloud Private)
IBM Cloud Private is an integrated, private cloud environment you can use to deploy and manage containerized cloud applications.

observer
An observer is a service that extracts resource information and inserts it into the Agile Service Manager database.

Agile Service Manager includes a configuration UI to help you configure and run observer jobs.

observer job
The access details for a target system are defined in an observer job, which is triggered to retrieve data.

Observer jobs are configured and run from the Observer Configuration UI, and can be long-running or transient. For example, the Network Manager Observer topology 'load' job is a one-off, transient job, while the Network Manager and Event Observer 'listen' jobs are long-running, which run until explicitly stopped, or until the Observer is stopped.

You can configure observer jobs manually by editing the configuration files for specific observer jobs, instead of using the Observer Configuration UI.

For the IBM Cloud Private version of Agile Service Manager, observer jobs are defined and run using Swagger.

observer job script
In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

provider
A provider is usually a single data source within the scope of a tenant.

Note: A provider's **uniqueId** property for a resource is unique only within the scope of a provider.

proxy The Agile Service Manager Nginx proxy server (nasm-nginx) manages access to all other Agile Service Manager micro-services by rewriting URLs.

resource
A resource is a node in an interconnected topology, sometimes also referred to as a vertex, or simply a node. It can be anything in a user-specific topology that has been designated as such, for example a hardware or virtual device, a location, a user, or an application.

scaling
You can scale up you system horizontally or vertically.

Horizontal scaling means increasing the replication factor of a particular service, and may also require adding additional hardware.

Vertical scaling means that you add more power (CPU or RAM) to an existing machine.

seed A seed is a single resource that has been chosen as the starting point of a topology. Once defined, the topology view is expanded one 'hop' at a time (the number of hops are configurable with a maximum of 30).

status Status is a property of one or more resources, and a single resource can have different types of status.

Each status can be in one of three states: open, clear or closed.

The status of a resource can be derived from events, in the case of the resource having been retrieved via the Event Observer, or it can be supplied when resources are posted to the topology service.

Swagger
Agile Service Manager uses Swagger for automated documentation generation and utilizes a Swagger server for each micro-service.

You can access and explore the REST APIs of the topology service and observers using Swagger via the proxy service.

tenant A tenant is represented by a globally unique identifier, its tenant ID.

The default tenant ID is: `cfd95b7e-3bc7-4006-a4a8-a73a79c71255`

topology
The arrangement of interconnected resources within a network, viewed in the Netcool Agile Service Manager UI.

More information:

You can find additional information on the topology service in the “Topology service reference” on page 257 section.

More detailed information on the topology screen elements are in the “Topology viewer reference” on page 293 section.

More information on Swagger is included in the documentation where appropriate. You can find a list of the default Swagger URLs and ports here: “Default Swagger URLs” on page 311

Chapter 2. Planning

This section helps you to plan your installation and use of Netcool Agile Service Manager by listing the minimum software and hardware requirements.

Hardware requirements

This section lists the minimum hardware requirements.

Your minimum hardware requirements are determined by the needs of the components of your specific Netcool Operations Insight solution. See the IBM Netcool Operations Insight Knowledge Center for more information: https://www.ibm.com/support/knowledgecenter/SSTPTP

For Agile Service Manager on IBM Cloud Private, see the relevant installation topic, as well as the Netcool Operations Insight topics pertaining to ICP deployment: https://www.ibm.com/support/knowledgecenter/SSTPTP_1.6.0/com.ibm.netcool_ops.doc/soc/integration/concept/int_deploying-on-icp.html

For its on-prem edition, the Agile Service Manager core components are deployed to a single server and the following physical or virtual hardware requirements must be met. Specifically, a number of Kernel parameters must be configured to optimize Cassandra and ElasticSearch, which run better when Swap is either disabled or the Kernel **vm.swappiness** parameter is set to 1 (in the `sysctl.conf` file). This setting reduces the Kernel's tendency to swap and should not lead to swapping under normal circumstances, while still allowing the whole system to swap under emergency conditions. See the following topic in the reference section for an example of a `sysctl.conf` file: “Example sysctl.conf file” on page 310

Table 3. Netcool Agile Service Manager Core hardware requirements

Requirement	Setting	Notes
CPU	48 cores	
Memory	64 GB	
Disk	500 GB	**Recommendations** Use disk arrays with redundancy, such as RAID10, with a minimum of 1000 IOPS. Have separate disks for the following services under the `$ASM_HOME/data` directory (by creating mount-points in your Operating System): • `$ASM_HOME/data/cassandra` • `$ASM_HOME/data/elasticsearch` • `$ASM_HOME/data/kafka` • `$ASM_HOME/data/zookeeper`

See the “Sizing reference” on page 313 topic for more details.

Software requirements

This section lists the minimum software requirements.

Netcool Agile Service Manager Core has the following requirements.

Table 4. Netcool Agile Service Manager Core software requirements

Requirement	Details
Operating system	Red Hat Enterprise Linux 7 (x86-64) Apply the latest updates.
Docker	Docker for Red Hat Enterprise Linux Version 1.12 Installation is described in the core installation topic. You can find more information about the Docker engine here: https://docs.docker.com/engine

The Netcool Agile Service Manager User Interface is deployed into an existing DASH instance that has been deployed as part of a Netcool Operations Insight installation. The UI has the following requirements.

Table 5. Netcool Agile Service Manager UI software requirements

Requirement	Details
Netcool Agile Service Manager Core	The UI component for Netcool Agile Service Manager can only be deployed once the core components have been installed.
WebSphere	WebSphere Application Server Version 8.5.5.4 or later
Java	IBM WebSphere SDK Java Technology Edition Version 7.0 or later
DASH	IBM Dashboard Application Service Hub (DASH) 3.1.2.1 or later
Netcool/OMNIbus probe for Message Bus	Version 8 or later
Netcool/OMNIbus XML Gateway	Version 9 or later

Chapter 3. Installing Agile Service Manager

Agile Service Manager is available as an IBM Cloud Private (ICP) as well as an on-prem deployment. The installation and configuration steps differ for these versions.

Tip: Both the ICP and on-prem versions of Agile Service Manager are installed together with Netcool Operations Insight. In ICP, Agile Service Manager and NOI must be installed into the same namespace.

Related concepts:

Chapter 2, "Planning," on page 9
This section helps you to plan your installation and use of Netcool Agile Service Manager by listing the minimum software and hardware requirements.

Related reference:

"Sizing reference" on page 313
The default deployment configuration will start three instances of Cassandra, Elasticsearch, Kafka and Zookeeper. This topic lists the compute and storage resources required for a default production deployment (size1). To ensure resiliency, you need a minimum of three worker nodes in your cluster with this configuration.

Installing and configuring on-prem

To install Netcool Agile Service Manager, you complete a number of prerequisites tasks. You then install the Netcool Agile Service Manager core components and observers, before deploying the user interface into DASH.

Installing the Netcool Agile Service Manager core services

The Netcool Agile Service Manager core application consists of several micro-services, which are provided as Docker containers. These are deployed onto a single server.

Before you begin

Updating your system: If you are updating an existing installation with the latest version of Agile Service Manager, you may already have the prerequisites in place. You should still ensure that you have the correct version of the prerequisites before upgrading your installation.

You must complete the following prerequisites before you can install the Netcool Agile Service Manager core applications.

1. Ensure that your Red Hat Enterprise Linux 7 (x86-64) operating system has the latest updates applied.
2. Ensure SELinux is disabled before performing the Netcool Agile Service Manager core application installation. To do so, edit the `/etc/selinux/config` file with a suitable editor and set `SELINUX=disabled`, before rebooting.
3. Enable the `rhel-x86_64-server-extras-7` and `rhel-7-server-optional-rpms` repositories so that the docker package can be installed. You can find more information on configuring the repository here: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/

System_Administrators_Guide/sec-Configuring_Yum_and_Yum_Repositories.html#sec-Setting_main_Options

4. Using the following commands, create the Docker group and add your current user to that group to enable non-root access to the Docker daemon.

   ```
   $ sudo groupadd docker
   $ sudo usermod -aG docker $USER
   ```

   More information on creating and administering Docker groups can be found here: https://docs.docker.com/engine/installation/linux/linux-postinstall/

   Important: For the group changes to take effect, you must refresh your terminal session.

   Tip: If you do not complete this step, you will either have to run the commands as the root user, or prefix your commands with the `sudo` command.

5. Obtain the Netcool Agile Service Manager core installation image from the Passport Advantage site, and extract it to a temporary directory. More information can be found in the download document here: http://www-01.ibm.com/support/docview.wss?uid=swg24043717

   Note: You need an IBM ID to access the download document.

About this task

When you install the Agile Service Manager components, they are loaded up automatically.

You install the core applications of Agile Service Manager and **only** the observers that you require. The Docker Observer is a requirement, as it supplies Agile System Manager health view data in the Topology Viewer. You should also install the Event Observer. In the unlikely event that you wish to install (and start up) all available observers, you must ensure that your system meets the minimum requirements listed here: Table 3 on page 9

After installing the core applications and the observers, you install the user interface.

Note: The example data for software versions or directories used here may differ from your own scenario.

Procedure

Prepare the installation files

1. Move all Agile Service Manager core and observer packages to the installation target host.
   - Copy only the observers you intend to deploy to the installation directory, or delete any unwanted observer packages after you have downloaded them. Remember that the Docker Observer is required.
   - Place the Agile Service Manager Base eAssembly and observer packages into the same directory.

     Important: To prevent unwanted observers being installed and thereby placing unnecessary strain on your infrastructure, ensure that this directory contains **only** the observers you wish to install.

Install Agile Service Manager core and observers

2. From the directory where you have placed the packages, install the Agile Service Manager core components and observers using the `yum install` command.

   Tip: While it is possible to specify each individual installation image, it is recommended that you perform a wildcard installation to ensure that all components are installed. Remember that you **must** ensure that only the observers you wish to install are present.

   ```
   sudo yum install nasm-*.rpm
   ```

   Yum will install Docker and all other `nasm-*` components as required, including all observers found in that directory. During the installation of the packages, the related Docker images are loaded. No data can be retrieved, however, until observer jobs are defined. Repeat this step for all the observers that you require, before moving on to the next step.

3. Required: During a first installation or during upgrades, you will be prompted to review and accept the license. You must do so **after installation has completed** using the following command:

   ```
   /opt/ibm/netcool/asm/bin/license-review.sh
   ```

   Note: If you do not complete this step and accept the license, Agile Service Manager will not start.

4. Optional: You can verify that the images have been loaded using the `docker images` command.

Start the Netcool Agile Service Manager services using Docker Compose

5. Add `/opt/ibm/netcool/asm/bin` to the **`$PATH`** variable.
6. Switch to the ASM_HOME directory (`$ASM_HOME=/opt/ibm/netcool/asm`), and then run the `docker-compose up -d` command. For example:

   ```
   $ export PATH=/opt/ibm/netcool/asm/bin:$PATH
   $ cd /opt/ibm/netcool/asm
   $ docker-compose up -d
   ```

   The first time this command is run, the containers will be created from the images. If the configuration is changed, this command will recreate the changed containers. The following is an example of system output while this happens:

   ```
   Creating network "asm_default" with the default driver
   Creating asm_zookeeper_1
   Creating asm_cassandra_1
   Creating asm_kafka_1
   Creating asm_topology_1
   Creating asm_event-observer_1
   Creating asm_itnm-observer_1
   Creating asm_openstack-observer_1
   ...
   ```

7. Verify the state of the containers.

   ```
   $ docker-compose ps
   ```

   The system output lists the state of all running services. The following example system output indicates that the Cassandra service is running (the **`State`** is **`Up`**):

   ```
   Name             Command             State
   ------------------------------------------------------------------------------
   ...
   inasmcore_cassandra_1   /bin/sh -c /opt/ibm/start- ...   Up
   ...
   ```

Results

You can view the service output on the Docker host 'logs' directory.

You can view the topology service API documentation in a web browser via its proxy rule. For example:

`https://<your-docker-host>/1.0/topology/swagger`

See the following table of the default ports and Swagger URLs for each service: "Default Swagger URLs" on page 311

What to do next

Next, you install the Netcool Agile Service Manager UI.

IBM Installation Manager

You use IBM Installation Manager to install the Netcool Agile Service Manager UI. Installation Manager is available for download from the IBM Fix Central website.

Before you begin

The recommended version of Installation Manager is 1.8.6.0.

Tip: If you are deploying Netcool Agile Service Manager as part of another IBM solution such as Netcool Operations Insight (NOI) you will already have IBM Installation Manager on your system.

You must have an IBM ID to download software from IBM Fix Central. You can register for an IBM ID at http://www.ibm.com.

You can find the IBM Installation Manager Knowledge Center at the following location: https://www.ibm.com/support/knowledgecenter/en/SSDV2W/im_family_welcome.html

About this task

The IBM Fix Central website offers two approaches to finding product files: **Select product** and **Find product**. The following instructions apply to the **Find product** option.

Procedure

1. Open the IBM Fix Central website at the following URL: http://www.ibm.com/support/fixcentral/
2. On the **Find product** tab:
   a. Enter `IBM Installation Manager` in the **Product selector** field.
   b. Select **1.8.6.0** from the **Installed Version** list.
   c. Select your intended host operating system from the **Platform** list and click **Continue**.
3. On the Identify Fixes page, choose **Browse for fixes** and **Show fixes that apply to this version (1.8.6.0)**. Click **Continue**.
4. On the Select Fixes page, select the installation file appropriate to your intended host operating system and click **Continue**.
5. When prompted, enter your IBM ID user name and password.

6. If your browser has Java enabled, choose the Download Director option. Otherwise, select the HTTP download option.
7. Start the installation file download. Make a note of the download location.

What to do next

Install Installation Manager (GUI, console, or silent installation).

Installing IBM Installation Manager (GUI or console)

You can install Installation Manager with a wizard-style GUI or an interactive console.

Before you begin

Take the following actions:

- Extract the contents of the Installation Manager installation file to a suitable temporary directory.
- Ensure that the necessary user permissions are in place for your intended installation, data, and shared directories.
- The console installer does not report required disk space. Ensure that you have enough free space before you start a console installation.

About this task

The initial installation steps are different depending on which user mode you use. The steps for completing the installation are common to all user modes and operating systems.

Installation Manager takes account of your current umask settings when it sets the permissions mode of the files and directories that it installs. Using Group mode, Installation Manager ignores any group bits that are set and uses a umask of 2 if the resulting value is 0.

Procedure

1. To install in Group mode:
   a. Use the id utility to verify that your current effective user group is suitable for the installation. If necessary, use the following command to start a new shell with the correct effective group:
      `newgrp group_name`
      We recommend using the icosgrp for Netcool Agile Service Manager.
   b. Use the umask utility to check your umask value. If necessary, change the umask value.
   c. Change to the temporary directory that contains the Installation Manager installation files.
   d. Use the following command to start the installation:

      GUI installation

      `./groupinst -dL data_location`

      Console installation

      `./groupinstc -c -dL data_location`

      Where *data_location* specifies the data directory. You must specify a data directory that all members of the group can access. Each instance of Installation Manager requires a different data directory

2. Follow the installer instructions to complete the installation. The installer requires the following input at different stages of the installation:

   GUI installation

   - In the first panel, select the `Installation Manager` package.
   - Read and accept the license agreement.
   - When prompted, enter an installation directory or accept the default directory.
   - Verify that the total installation size does not exceed the available disk space.
   - When prompted, restart Installation Manager.

   Console installation

   - Read and accept the license agreement.
   - When prompted, enter an installation directory or accept the default directory.
   - If required, generate a response file. Enter the directory path and a file name with a `.xml` extension. The response file is generated before installation completes.
   - When prompted, restart Installation Manager.

Results

Installation Manager is installed and can now be used to install the Netcool Agile Service Manager UI.

What to do next

If required, add the Installation Manager installation directory path to your `PATH` environment variable.

Installing the Netcool Agile Service Manager UI using the Installation Manager

The Netcool Agile Service Manager user interface is installed into an existing DASH installation, and then configured to communicate with the Netcool Agile Service Manager core services.

Before you begin

Important:

If you have an installation of Netcool Agile Service Manager that was installed using the legacy, pre-IBM Installation Manager process, you must uninstall that version using the legacy uninstall process first. See the following topic for detailed instructions: "[BETA legacy] Uninstalling the Netcool Agile Service Manager UI" on page 33

Updating your system: If you are updating an existing Agile Service Manager UI rather than installing a new version, choose **Update** instead of **Install** when completing this step of the procedure. Note that even if you already have the prerequisites in place, you should ensure that you have the correct version of the prerequisites before upgrading your installation.

You must complete the following prerequisites before you can install the Netcool Agile Service Manager user interface.

1. Ensure that Netcool Agile Service Manager core has been installed and is running.
2. Ensure that WebSphere Application Server Version 8.5.5 has been installed. Follow the IBM Knowledge Center instructions here: https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/welcome_ndmp.html
3. Ensure that a compatible version of the IBM WebSphere Java SDK has been installed (for example together with WebSphere 8.5.5.15) The Java SDK should be at least Version 7.1. Follow the IBM Knowledge Center instructions here: https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.installation.nd.doc/ae/tins_installation_jdk7_gui.html
4. Ensure that IBM Dashboard Application Service Hub (DASH) 3.1.2.1 or later has been installed. Follow the IBM Knowledge Center instructions for installing Jazz for Service Management here: https://www.ibm.com/support/knowledgecenter/en/SSEKCU_1.1.3.0/com.ibm.psc.doc/install/psc_c_install.html
5. Ensure that IBM Netcool Agile Service Manager Core is accessible on your network from the machine that is hosting DASH.
6. Obtain the Netcool Agile Service Manager UI installation image from the Passport Advantage site, and extract it to a temporary directory. More information can be found in the download document here: http://www-01.ibm.com/support/docview.wss?uid=swg24043717

   Note: You need an IBM ID to access the download document.
7. The IBM Installation Manager analyzes existing installations to determine defaults, which it then presents to you during installation. To verify these defaults and change them if necessary, ensure you have the following environment information about your installation and the DASH environment to hand.

   DASH_PROFILE

   The *DASH_Profile* variable describes the location of the application server profile that is used for Dashboard Application Services Hub. This location is in the `/profile/` subdirectory of the Jazz for Service Management home directory.

   The default root user location is `/opt/IBM/JazzSM/profile`

   The default non-root user location is `/home/<nonrootuser_name>/IBM/JazzSM/profile`

   DASH_HOME

   The *DASH_HOME* variable describes the (configurable) location where Dashboard Application Services Hub is installed.

   The default root user location is `/opt/IBM/JazzSM/ui`

   The default non-root user location is `/home/<nonrootuser_name>/IBM/JazzSM/ui`

   WAS_HOME

   The *WAS_HOME* variable describes the (configurable) location where WebSphere Application Server is installed.

   The default root user location is `/opt/IBM/WebSphere/AppServer`

The default non-root user location is `/home/<nonrootuser_name>/IBM/WebSphere/AppServer`

Note: For more information on DASH and WAS environment variables, see the following topic in the Jazz for Service Management Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSEKCU_1.1.2.1/com.ibm.psc.doc/ref/psc_r_pathnames.html#psc_r_pathnames__tip_root

NCHOME
The Netcool home location.

Usually found at `/opt/IBM/netcool/gui`

Proxy Service Host
The host address (host name or IP address) for the Agile Service Manager proxy service.

Proxy Service Port
The port number for the Agile Service Manager proxy service.

Attention: The default is 443 (previously 80).

Tenant ID
This is the GUID that will be used to access any data associated with your tenant.

The default ID is `cfd95b7e-3bc7-4006-a4a8-a73a79c71255`

Tip: You can specify another GUID, but you must ensure that this is used in all API calls when creating data resources.

Core Username and Password
The name and password of the user used to authenticate with the Agile Service Manager core services.

Authentication is enabled by default.

The default values for the user name and password are **asm** and **asm**, respectively.

About this task

The Netcool Agile Service Manager UI install bundle contains the Installation Manager zip archive `com.ibm.itsm.topology.ui`

You add the Netcool Agile Service Manager UI to an existing DASH installation and configure the application to communicate with the previously installed core application.

Important: Ensure that you are logged in as the same user who installed DASH.

The steps for starting Installation Manager differ depending on the user mode in which it was installed. The steps for installing with the Installation Manager console are common to all user modes and operating systems. Take note of the following information about permissions on the supported operating systems:

- The Installation Manager takes account of your current umask settings when it sets the permissions mode of the files and directories that it installs.
- If you use Administrator mode or Nonadministrator mode and your umask is 0, Installation Manager uses a umask of 22.

- If you use Group mode, Installation Manager ignores any group bits that are set and uses a umask of 2 if the resulting value is 0.

Note: It is recommended to use Install Manager Group Mode for installation. Respond to each Installation configuration option to ensure it matches your pre-defined Installation Information Checklist.

Procedure

1. Change to the `/eclipse` subdirectory of the Installation Manager Group installation directory and use the following command to start Installation Manager:

   ```
   ./IBMIM
   ```

2. From the Installation Manager Main Menu, select **Preferences**.
3. In the Preferences menu, select **Repositories**.
4. In the Repositories menu, select **Add Repository**.
5. Enter the path to `repository.config` in the extracted directory, and return to the Main Menu.
6. In the main Installation Manager window, click **Install** for a fresh installation, or **Update** when upgrading an existing UI installation. If updating, the installation process will detect and use existing configuration settings where possible.

   Upgrade Note: During an upgrade installation a connection settings panel will be displayed, including the values entered during your previous installation. **Always double-check all values.**
7. Select the UI package (the latest version of IBM Netcool Agile Service Manager), and click **Next**.
8. Complete the following installation steps:
   a. On the Installation Manager Licenses tab, read and accept the license agreement, then click **Next**. The installation process moves onto the Location tab.
   b. Select the IBM Netcool GUI Components package group and accept the default `/opt/IBM/netcool/gui` location, then click **Next**. The installation process moves onto the Features tab, which displays your selected version of Netcool Agile Service Manager, as well as the features you can install.
   c. Select **IBM Netcool Agile Service Manager User Interface**, then click **Next**.
   d. Enter the required information for the WebSphere Application Server, then click **Next**.
   e. Enter the tenant ID, host name and port of the Agile Service Manager proxy service, as well as the username and password for the main Agile Service Manager backend (core) services, then click **Next**.

      Tip: When you enter a username or password for the backend services (which would be `asm` for both username and password for a standard Agile Service Manager installation), their values will be stored in the Agile Service Manager UI application settings file (`application.yml`). By default the password is stored in this file in plain text. To improve security, you can encrypt the password once the installation is complete. For more information, see the core services configuration topics, which you can access from the related links.

Note: You can test the connection details entered here (using **Test Connection**) to ensure that the UI can connect to the Agile Service Manager proxy service on the port specified.

f. Click **Install** to complete the installation, then **Finish** to exit the installer.

Results

The IBM Installation Manager installs Netcool Agile Service Manager into your existing DASH installation, and then restarts DASH.

Troubleshooting Tip:

The following installation errors can occur when system resources are insufficient:

```
ERROR:
com.ibm.itsm.topology.ui.install.InstallException: Failed to stop and
restart the server server1.
CRIMA1076E: Error executing the /opt/IBM/jazz/ui/bin/consoleDeregister.sh
command: status=1.
```

Workaround

1. Stop Agile Service Manager and DASH.
2. Using the Installation Manager, uninstall the Agile Service Manager UI components.
3. Using the command line, remove all remaining files and directories, using **one** of the following commands (depending on your directory structure):

   ```
   rm -rf /opt/IBM/gui/inasm
   ```

   ```
   rm -rf /opt/ibm/netcool/gui/inasm
   ```
4. Using the Installation Manager, complete a fresh installation of the Agile Service Manager UI components.

What to do next

Next, you configure DASH user roles to allow users to access the Netcool Agile Service Manager UI.

You can also improve security by configuring authentication for UI access to the core services.

The Agile Service Manager UI communicates with the proxy service via HTTPS (TLS) using a default SSL trust store. You can customize the secure communication between the Agile Service Manager UI and the proxy service by changing the trust store password, updating or changing the trust store certificate, or using a custom trust store file instead of the one provided.

Tip: You modify the connection parameters for the Agile Service Manager proxy service by editing the application settings file (`$NCHOME/inasm/etc/application.yml`).

Related concepts:

"Configuring SSL between the UI and the proxy service" on page 187
The Agile Service Manager UI communicates with the proxy service via HTTPS (TLS). The UI uses a default SSL trust store containing the proxy service certificate to encrypt the communications between them.

Related tasks:

"Configuring DASH user roles"
You configure DASH user roles so that users can use the Netcool Agile Service Manager UI. This task is the same for both on-prem and ICP deployments of Agile Service Manager.

"Configuring core services authentication" on page 183
To customize secure access to the core services, you can change the default authentication method the UI uses when connecting to core services from basic authentication to LDAP. You can also encrypt the password and generate a new password encryption key.

Configuring DASH user roles

You configure DASH user roles so that users can use the Netcool Agile Service Manager UI. This task is the same for both on-prem and ICP deployments of Agile Service Manager.

About this task

You can assign the following DASH user roles to users:

inasm_operator
A user with the inasm_operator role can access the Netcool Agile Service Manager UI, and use it to search for and visualize the resources in the Netcool Agile Service Manager core application.

inasm_editor
The same as for inasm_operator.

In addition, a user with the inasm_editor role can add comments to resources from the Topology Viewer Context (right-click) menu. (A user with the inasm_operator role can view comments, but not add new ones.)

inasm_admin
The same as for inasm_editor.

In addition, a user with the inasm_admin role has access to a number of administrator tools, where they can define custom UI elements for the Topology Viewer. See the "Customizing UI elements" on page 191 topic for more information.

To configure DASH user roles, you must log into DASH with admin user credentials.

Tip: You can also assign roles to a user indirectly by assigning them to a group of which the user is a member.

Procedure

1. As the admin user, log into your DASH web application.

   For on-prem
   If you have used the default root location of `/ibm/console`, use the following logon URL:

   `https://<DASH-HOST>:<DASH-PORT>/ibm/console/logon.jsp`

   For ICP

   You login to the Agile Service Manager ICP installation using a URL of the following format (example):

   `https://netcool.noi.icp-master.<your_host>/ibm/console`

Where *noi* is the Netcool Operations Insight Helm release name. Use the following command to retrieve the DASH URL:

```
helm status NOI helm release name --tls
```

2. Select **Console Settings** from the DASH menu.
3. Select **User Roles** from the Console Settings menu (under the Roles heading).
4. Select the **User Roles** tab, and then click **Search**. Known users are displayed in the Search results table.
5. **For each user** requiring access to the Netcool Agile Service Manager UI, perform the following actions:
   a. Click the required user ID in the Search results table. All roles that are available to the selected user are displayed in the Available Roles table.
   b. Select the required roles, as appropriate.
   c. Click **Save**.

Results

Once you have saved your changes, the user role changes take effect. All users with their newly assigned roles are now able to log into DASH and access the Netcool Agile Service Manager UI. From there, users can search and visualize resources from the Netcool Agile Service Manager topology service.

Remember: You can also assign roles to a user by assigning them to a group to which the user belongs.

Editing the application settings file

You can modify the connection parameters for the Agile Service Manager proxy service, as well as the SSL communication between the UI and the proxy service, at any time after installation by editing the application settings file (`$NCHOME/inasm/etc/application.yml`).

About this task

You can change the host name, port number, tenant ID and other proxy service settings. If you edit the `application.yml` file, you must restart DASH before the changes can take effect.

Note: From Agile Service Manager Version 1.1.4: Communication between the UI and the proxy service is encrypted using SSL (TLS). To change the default SSL parameters, you change the trust store settings in the application.yml file.

The settings are stored in the Agile Service Manager UI application configuration file, which is located at `$NCHOME/inasm/etc/application.yml`

Example application config file location

```
/opt/IBM/netcool/gui/inasm/etc/application.yml
```

Important: If password encryption is being set to `true`, then **all** passwords (SSL trust store passwords **and** proxy service passwords) in the `application.yml` file must be encrypted.

Procedure

Edit the application configuration file

1. Open the application configuration file using an appropriate editor.

2. Edit the settings for the Agile Service Manager proxy service, such as:
   - Host name
   - Port number
   - Tenant ID
   - User name
   - Password
3. Edit the SSL trust store settings for secure communication between the UI and the proxy service (trust store path, password and type).
4. If authentication is enabled in the Agile Service Manager proxy service and you have therefore specified a password in the `application.yml` file, set the passwordEncryption property to true or false, as required. For more information on password encryption, see the related topics.

Restart DASH to allow the changes to take effect.

5. To stop the DASH server, run `<DASH_PROFILE>/bin/stopServer.sh server1`
6. Once stopped, start the DASH server: `<DASH_PROFILE>/bin/startServer.sh server1`

Related tasks:

"Configuring core services authentication" on page 183
To customize secure access to the core services, you can change the default authentication method the UI uses when connecting to core services from basic authentication to LDAP. You can also encrypt the password and generate a new password encryption key.

"Encrypting the password for UI access to core services" on page 184
You can encrypt the password used for connecting to the Agile Service Manager services using the `encrypt_password` tool located in the `INASM_UI_HOME/bin` directory.

"Generating a new password encryption key" on page 186
You can generate a new password encryption key using the `generate_key` tool located in the `INASM_UI_HOME/bin` directory. You can then use that new key file to encrypt passwords.

Deploying the XML Gateway for Event Observer

Agile Service Manager resource status can be generated from Omnibus events. You must configure the IBM Tivoli Netcool/OMNIbus XML Gateway for Message Bus to post XML events to the Event Observer.

Before you begin

Obtain the Netcool/OMNIbus XML Gateway from the Passport Advantage Online website. For more information, see the following IBM download document: https://www-304.ibm.com/support/docview.wss?uid=swg21665219

For more information on configuring the XML gateway, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/gateways/xmlintegration/wip/concept/xmlgw_intro.html

For additional gateway configuration information, see the following IBM developerWorks discussion: https://developer.ibm.com/answers/questions/256154/how-is-the-xml-message-bus-probe-and-gateway-confi.html

Important: Ensure you have all the required Netcool/OMNIbus Object Server information available before you install and configure the gateway.

Tip: Netcool/OMNIbus events can be generated from Agile Service Manager status. You configure the Netcool/OMNIbus Message Bus probe to receive status updates from Agile Service Manager, and generate corresponding Netcool/OMNIbus events. This ensures that event views across the Netcool Agile Service Manager Topology Viewer and the Netcool/OMNIbus Event Viewer remain synchronized, as depicted in this diagram. See the related links for more information.

About this task

Before you can define a job for the Event Observer, you must install and configure the IBM Tivoli Netcool/OMNIbus XML Gateway for Message Bus to post XML events to the Event Observer.

For the XML gateway to post XML events to the Event Observer, you must edit the following files as a minimum:

XML Gateway properties file
: If this file does not exist, you must create it in the `$OMNIHOME/etc` directory.

  For example, the XML Gateway properties file for a gateway called G_ASM would be `$OMNIHOME/etc/G_ASM.props`

  You define a number of properties, such as the name of the Netcool/OMNIbus Object Server, in the XML Gateway properties file.

  You also reference the transformers XML file and the transport properties file here.

XML Gateway transport properties file
: The file name of the XML Gateway transport properties file must match the one referenced in the XML Gateway properties file.

  Here you define as a minimum the URL of the Event Observer to which XML events are posted, the batch header and footer, the maximum number of events in a single batch, the maximum waiting period before sending the events, and access to the HTTPS (TLS) truststore.

  Default location and name: `$OMNIHOME/java/conf/asm_httpTransport.properties`

XML Gateway transformer XML file
: The file name of the XML Gateway transformer XML file must match the one referenced in the XML Gateway properties file.

  Here you define as a minimum the URL of the Event Observer to which XML events are posted.

  Default location and name: `$OMNIHOME/java/conf/asm_Transformers.xml`

Events are received in batch mode input format.

Batch mode
: The gateway is configured to batch multiple events into a single XML document.

  The Event Observer URL must be configured as the gateway endpoint URL as follows: `/1.0/event-observer/netcool/list`

The format consists of one or more rows from the Netcool/OMNIbus Object Server's **alerts.status** table, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:netcoolEventList  xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
 <tns:netcoolEvent type="update" xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
  <tns:netcoolField name="Serial" type="integer">1</tns:netcoolField>
  <tns:netcoolField name="Node" type="string">sol8-build2.hursley.ibm.com</tns:netcoolField>
  <tns:netcoolField name="LastOccurrence" type="utc">2017-02-14T08:16:48+0100</tns:netcoolField>
  ....
 </tns:netcoolEvent>
  <tns:netcoolEvent type="update" xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool">
  <tns:netcoolField name="Serial" type="integer">2</tns:netcoolField>
  <tns:netcoolField name="Node" type="string">sol8-build2.hursley.ibm.com</tns:netcoolField>
  <tns:netcoolField name="LastOccurrence" type="utc">2017-02-14T08:16:48+0100</tns:netcoolField>
  ....
 </tns:netcoolEvent>
 ...
</tns:netcoolEventList>
```

The Event Observer derives the status of resources from individual fields of the event.

Table 6. General event state rules

State	Meaning	Netcool/OMNIbus event mapping
closed	An active issue, may require attention	Active event with Severity > 0
open	Current state, working as expected	Cleared event with Severity = 0
clear	No longer relevant	Deleted event

Table 7. Use of Netcool/OMNIbus alerts.status event fields by Agile Service Manager

alerts.status fields	Use by Agile Service Manager
Agent	Provider name for events generated from Agile Service Manager
AlertGroup	Type of Agile Service Manager event
Class	45111 for Agile Service Manager events (should be mapped in alerts.conversions)
Customer	TenantId for events generated from Agile Service Manager
EventId	Status [type] for events generated from Agile Service Manager
Identifier	Determines the type of status, populating the status field
LastOccurrence	Used for the observedTime of open events
LocalPriObj	Resource lookup
LocalRootObj	Resource lookup
Manager	Observer name for events generated from Agile Service Manager
Node	Resource lookup
NodeAlias	Resource lookup
ServerName	Used to generate the unique eventId
ServerSerial	Used to generate the unique eventId
Severity	Severity 0 events represent a clear state
StateChange	Used for the observedTime of clear events
Summary	Used for the status description, shown in the UI

Table 7. Use of Netcool/OMNIbus alerts.status event fields by Agile Service Manager (continued)

alerts.status fields	Use by Agile Service Manager
Type	Only Type 1 (Problem), Type 13 (Information) and Type 20 (ITMProblem) events are processed. All others are ignored.

Table 8. Netcool/OMNIbus event data mapped onto Topology Service status

Topology Service status field	Netcool/OMNIbus source
description	alerts.status Summary
eventId	<ServerName>/<ServerSerial>
eventManager	"netcool"
observedTime	closed - time event received by observer clear - alerts.status StateChange open - alerts.status ObservedTime
severity	alerts.status Severity
state	closed - deleted events clear - Severity 0 events open - none of the above
status	alerts.status Identifier

Procedure

1. Download and install the Netcool/OMNIbus XML Gateway using the IBM Installation Manager.

   Remember: For up-to-date information on the version of the XML Gateway required, see "Software requirements" on page 10.

   Location

   The default $NCHOME install location is `/opt/IBM/tivoli/netcool` and the $OMNIHOME install location is `$NCHOME/omnibus`.

   Tip: Export OMNIHOME as an environment variable, as it is repeatedly used in the scripts.

   Standard gateway configuration

   You must create a `$NCHOME/etc/omni.dat` entry for the gateway (which in these examples is assumed to be **G_ASM**):

   ```
   [G_ASM]
    {
    Primary: nasm-test1 4300
    }
   ```

   Run **`$NCHOME/bin/nco_igen`**

   Generate a key file with nco_keygen.

2. Use the following information to configure the XML Gateway properties file.

   You create and/or edit the XML Gateway properties file: `$OMNIHOME/etc/<your gateway>.props` and then define at least the following properties:

- The name of the Netcool/OMNIbus ObjectServer
- The name of the transport properties file
- The name of the transformer XML file

The following sample code is for a `$OMNIHOME/etc/G_ASM.props` gateway properties file, retrieving data from the `AGG_V` ObjectServer via the `G_ASM` gateway.

```
# Standard properties
Gate.Reader.Server  : 'AGG_V'

# Properties defining XML messages over HTTP
Gate.MapFile:                    '$OMNIHOME/gates/xml/asm_xml.map'
Gate.StartupCmdFile:             '$OMNIHOME/gates/xml/xml.startup.cmd'
Gate.Reader.TblReplicateDefFile: '$OMNIHOME/gates/xml/asm_xml.reader.tblrep.def'
Gate.XMLGateway.TransformerFile: '$OMNIHOME/java/conf/asm_transformers.xml'
Gate.XMLGateway.TransportFile:   '$OMNIHOME/java/conf/asm_httpTransport.properties'
Gate.XMLGateway.TransportType:   'HTTP'

# The event observer requires the timestamp in this format, including the timezone
Gate.XMLGateway.DateFormat      : 'yyyy-MM-dd\'T\'HH:mm:ssZ'

# To flush events to the gateway from the object server at 5s intervals, use this
Gate.Reader.IducFlushRate : 5

####################################################
# Security credentials required for the proxy service
# For full FIPS compliance, alter the java.security file as per the Omnibus documentation
####################################################
# This algorithm must be AES_FIPS
ConfigCryptoAlg: 'AES_FIPS'
# Secure key file generated using nco_keygen
ConfigKeyFile:   '/opt/IBM/netcool/core/omnibus/etc/crypto.key'
```

Important: Do **not** use the $OMNIHOME variable in ConfigKeyFile.
Example mapping (minimum fields required)

Note: The name of the gateway map file must match the one specified by the `Gate.MapFile` property in the gateway properties file.

```
CREATE MAPPING StatusMap
(
  'Agent'          = '@Agent',
  'AlertGroup'     = '@AlertGroup',
  'Class'          = '@Class',
  'Customer'       = '@Customer',
  'EventId'        = '@EventId',
  'Identifier'     = '@Identifier',
  'LastOccurrence' = '@LastOccurrence',
  'LocalPriObj'    = '@LocalPriObj',
  'LocalRootObj'   = '@LocalRootObj',
  'Manager'        = '@Manager',
  'Node'           = '@Node',
  'NodeAlias'      = '@NodeAlias',
  'ServerName'     = '@ServerName',
  'ServerSerial'   = '@ServerSerial',
  'Severity'       = '@Severity',
  'StateChange'    = '@StateChange',
  'Summary'        = '@Summary',
  'Type'           = '@Type'
);
```

3. Use the following information to configure the `asm_httpTransport.properties` file.

Note: The name of the gateway transport properties file must match the one specified by the `Gate.XMLGateway.TransportFile` property in the gateway properties file.

The gateway transport properties file (in these examples $OMNIHOME/java/conf/asm_httpTransport.properties) must specify at least the following properties, as shown in the example:

- The user's authentication credentials.
- The batch header and footer, exactly as shown in the example.
- The size and flush time, which specify the maximum number of events in a single XML batch file, and the maximum wait, in seconds, before sending the events.
- The proxy service username and password. Encrypt the proxy service password (and optionally the username):

  ```
  nco_aes_crypt -c AES_FIPS -k /opt/IBM/netcool/core/omnibus/etc/crypto.key <password>
  ```

- Add the username and password to the asm_httpTransport.properties file, for example:

  ```
  username=asm
  password=@44:9WxiH51VqMNHNYOLvoShaXO01KwBLqXtGqtB/ZGCYPo=@
  ```

  Tip: You only edit the java security file for FIPS compliance.

- For gateway access to the truststore, you need to complete the following steps:

  a. Create a truststore from the ASM CA certificate, and copy it to the Netcool/OMNIbus host (if different).

  ```
  keytool -import \
          -alias asm-ca \
          -file $ASM_HOME/security/asm-ca.crt \
          -keystore gateway_truststore.jks \  <---- this file needs to go in the gw config
          -storetype JKS \
          -noprompt
  ```

  While running the command, you are prompted for a password.

  b. Add the truststore and password to the Gateway transport properties file. When completed, the gateway transport properties file should contain the following:

  – trustStore=/fullPath/gateway_truststore.jks
  – trustStorePassword={passwordGivenInPreviousStep}

  In the following example you are copying gateway_truststore.jks from the Agile Service Manager server:

  ```
  $ grep ^trust /opt/IBM/netcool/core/omnibus/java/conf/asm_httpTransport.properties
  trustStore=/opt/ibm/netcool/asm/security/gateway_truststore.jks
  trustStorePassword=changeit
  ```

Optionally, you can also define:

- The timeout, which is the amount of time in seconds that an http client waits before aborting the connection.
- The retry limit, which is the number of times a http client tries to connect.
- The retry wait time, which is the amount of time (in seconds) an http client waits before attempting to reconnect.

4. Use the following information to configure the asm_Transformers.xml file.

   The gateway transformer XML file ($OMNIHOME/java/conf/asm_Transformers.xml) must specify at least the URL of the Event Observer (endpoint), to which XML events are posted.

   Note: The name of the gateway transformer XML file must match the one specified by the Gate.XMLGateway.TransformerFile property in the gateway

properties file.
In the following example, the `your host` part of the URL specified in `endpoint` will be specific to your installation.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformers
    xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool/transformer"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <tns:transformer name="netcoolEvents" type="northbound"
endpoint="https://<your-asm-host>/1.0/event-observer/netcool/list"
className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">
        <tns:property name="xsltFilename" type="java.lang.String"
value="${OMNIHOME}/java/conf/netcoolstripxmlheader.xsl"
description="XSLT file for converting Netcool events to NC events"/>
    </tns:transformer>
</tns:transformers>
```

5. Optional: The gateway can be configured to filter Netcool/OMNIbus events and send only a subset to the Event Observer.

   Note: The name of the table replication definition file must match the one specified by the `Gate.Reader.TblReplicateDefFile` property in the gateway properties file.
   To improve performance and prevent unnecessary events from being displayed in the topology viewer, you can filter events by type, and then refine these further by extracting only the fields of interest.

   To include only problem (type 1), information (type 13), and ITMProblem (type 20) event types, and exclude Netcool/OMNIbus self-monitoring events (class 99999), use the following code:

   ```
   asm_xml.reader.tblrep.def: |-
   REPLICATE ALL FROM TABLE 'alerts.status'
   USING MAP 'StatusMap'
   FILTER WITH 'Type IN (1, 13, 20) AND Class != 99999';
   ```

Results

The Netcool/OMNIbus XML Gateway is ready to supply Netcool/OMNIbus event data to the Event Observer.

What to do next

Next, you configure the Event Observer job.

Remember: To keep event views across the Netcool Agile Service Manager Topology Viewer and the Netcool/OMNIbus Event Viewer synchronized, you must configure the Netcool/OMNIbus Message Bus probe to receive status updates from Agile Service Manager, and generate corresponding Netcool/OMNIbus events.

Related tasks:

"Deploying the Netcool/OMNIbus probe for Message Bus" on page 30
Netcool/OMNIbus events can be generated from Agile Service Manager status. You use the IBM Tivoli Netcool/OMNIbus Message Bus probe (nco_p_message_bus) to receive status from Agile Service Manager and generate corresponding events in Netcool/OMNIbus.

"Defining Event Observer jobs" on page 123
You use the Event Observer to get events from Netcool/OMNIbus, via the XML Gateway, into the Netcool Agile Service Manager topology service.
Netcool/OMNIbus events can also be generated from Agile Service Manager status via the Netcool/OMNIbus Message Bus probe. The Event Observer is installed as part of the core installation procedure.

Deploying the Netcool/OMNIbus probe for Message Bus

Netcool/OMNIbus events can be generated from Agile Service Manager status. You use the IBM Tivoli Netcool/OMNIbus Message Bus probe (nco_p_message_bus) to receive status from Agile Service Manager and generate corresponding events in Netcool/OMNIbus.

Before you begin

Obtain the Netcool/OMNIbus Message Bus probe from the Passport Advantage Online website. For more information, see the following IBM download document: http://www-01.ibm.com/support/docview.wss?uid=swg21970413

For more information on configuring the probe, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/message_bus/wip/concept/messbuspr_intro.html

For information on using the probe and the gateway as a single implementation, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/message_bus/wip/concept/messbuspr_integration_intro.html

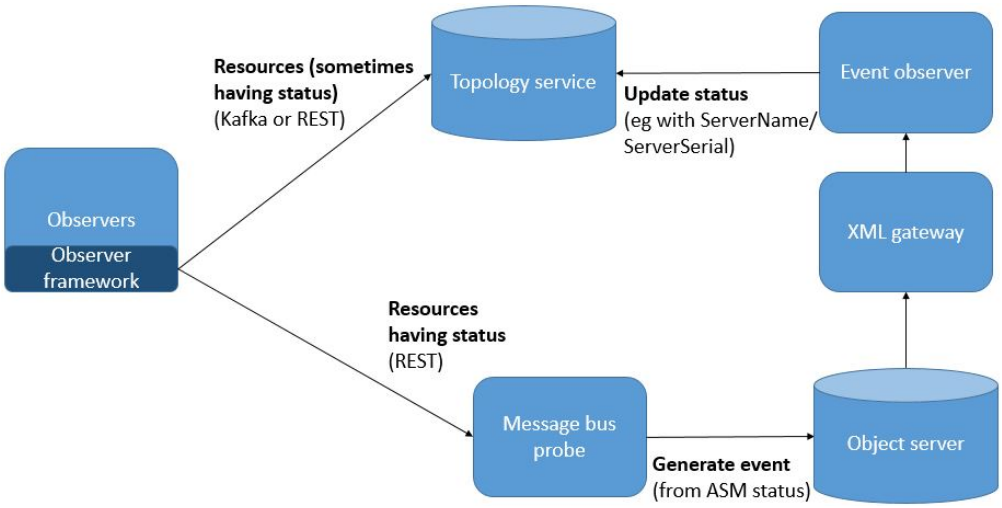
Important: Ensure you have all the required Netcool/OMNIbus Object Server information available before you install and configure the gateway.

Before setting up the probe, you must configure the gateway to post XML events to the Event Observer, as described in the following topic: “Deploying the XML Gateway for Event Observer” on page 23

About this task

The Netcool/OMNIbus Message Bus probe must be configured to receive status from Agile Service Manager in JSON format via HTTP, and generate corresponding events in the Netcool/OMNIbus Event Viewer. These events are then fed back to the Agile Service Manager via the Netcool/OMNIbus XML Gateway, which updates the Agile Service Manager status via the Event Observer with the eventId.

The following diagram depicts how the Netcool/OMNIbus Message Bus probe and the XML Gateway work together with the Agile Service Manager Event Observer to keep the event status between Agile Service Manager and Netcool/OMNIbus synchronized.



Specifically, this diagram shows how the data flow from Agile Service Manager status generates Netcool/OMNIbus events. When Netcool/OMNIbus is the source of events, however, the data flow from the Event Observer to the topology service not only updates the status eventId with ServerName/ServerSerial, but generates the status itself.

Procedure

1. Download and install the Netcool/OMNIbus probe for Message Bus using the IBM Installation Manager.

   Location

   The default OMNIHOME install location is `/opt/IBM/tivoli/netcool/omnibus`

2. Use the following information to configure the probe properties file so that it can receive Agile Service Manager status.

   Probe properties file

   Create and edit the probe property file.

   In the following example, a non-default property file is used, which requires the -propsfile option when running the probe.

   ```
   cd $OMNIHOME/probes/linux2x86/
   cp message_bus.props asm_message_bus.props
   ```

   Edit the asm_message_bus.props file as in the following example:

   ```
   # Tell the probe to expect json over REST
   MessagePayload  : 'json'
   TransformerFile : ''
   TransportFile   : '$OMNIHOME/java/conf/probe_httpTransport.properties'
   TransportType   : 'HTTP'

   # Tell the probe how to parse the json payload, such that each member
   of its variable-length
   # _status array is processed as a separate message, with top-level
   properties also included
   MessageHeader                   : 'json'
   MessagePayload                  : 'json._status'

   # standard probe properties
   MessageLog                      : '$OMNIHOME/log/asm_probe.log'
   RulesFile                       : '$OMNIHOME/probes/linux2x86/
   asm_message_bus.rules'
   ```

3. Use the following information to configure the probe transport file.

Probe transport file

Create and edit the probe transport file.

The name of the probe transport file must match the name given in the probe properties, in this example 'probe_httpTransport.properties'

Create a new file if necessary:

```
cd $OMNIHOME
cp java/conf/httpTransport.properties java/conf/probe_httpTransport.properties
```

This file needs to specify at least the URL of the probe, where it will accept JSON status as input; for example:

```
serverPort=http:18080
```

This port number is required when registering the probe URL.

4. Copy the probe rules file included in the Agile Service Manager installation from the following location: `$ASM_HOME/integrations/omnibus/asm_message_bus.rules` to the location specified in the probe properties file RulesFile property in step 2 (in this case `$OMNIHOME/probes/linux2x86/asm_message_bus.rules`).
5. Start the probe using Netcool/OMNIbus procedures, as documented in the Netcool/OMNIbus probes and gateways knowledge center.
6. Register the probe URL to which status is exported using the port configured in step three. Run the script included in `$ASM_HOME/bin`, as in the following example:

   ```
   ./topology_service_register_probe.sh -url http://<your.hostname>:18080
   ```

   See the Probe for Message Bus reference topic for an example ASM_EVENT_SINK management artifact.
7. Deploy the Netcool/OMNIbus 'asm_resource_deletion' trigger into Netcool/OMNIbus by loading the `$ASM_HOME/omnibus/integrations/asm.sql` file into the Netcool/OMNIbus Object Server. For example:

   ```
   ./bin/nco_sql -U root -S ASM_AGG_P -input /opt/ibm/netcool/asm/integrations/omnibus/asm.sql
   ```

   Tip: You may receive an `asm_resource_deletion` error message, as in the following example, when Agile Service Manager event triggers are first created in Netcool/OMNIbus. This warning is caused by the process that first removes existing triggers before recreating them. During its first run it will not encounter any existing triggers, and so will display an error message. You can ignore this message.

   ```
   ERROR=Object not found on line 20 of statement
   '-------------------------------------------------------------------------...', at
   or near 'asm_resource_deletion'
   ERROR=Object not found on line 1 of statement 'drop table
   alerts.asm_deleted_resources;...', at or near 'asm_deleted_resources'
   (0 rows affected)
   (0 rows affected)
   ```

Results

The Netcool/OMNIbus Message Bus probe is ready to receive status from Agile Service Manager, and then pass it on to the Netcool/OMNIbus Event Viewer.

What to do next

Having configured the Netcool/OMNIbus probe, and having previously configured the Netcool/OMNIbus gateway, you still need to configure the Event

Observer job. After that events across Agile Service Manager and Netcool/OMNIbus can be synchronized, as depicted in this diagram.

Related tasks:

"Defining Event Observer jobs" on page 123
You use the Event Observer to get events from Netcool/OMNIbus, via the XML Gateway, into the Netcool Agile Service Manager topology service. Netcool/OMNIbus events can also be generated from Agile Service Manager status via the Netcool/OMNIbus Message Bus probe. The Event Observer is installed as part of the core installation procedure.

"Deploying the XML Gateway for Event Observer" on page 23
Agile Service Manager resource status can be generated from Omnibus events. You must configure the IBM Tivoli Netcool/OMNIbus XML Gateway for Message Bus to post XML events to the Event Observer.

Related information:

https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/common/Probes.html

[BETA legacy] Uninstalling the Netcool Agile Service Manager UI

If you have installed an earlier version of the Netcool Agile Service Manager UI using the legacy installation process, that is, the process used **before** IBM Installation Manager, you must use the legacy uninstall process before installing a new version.

Before you begin

Before uninstalling the Netcool Agile Service Manager UI, check that the uninstall scripts are present, and ensure that you have the installation user credentials for the Dashboard Application Service Hub (DASH).

To uninstall the Netcool Agile Service Manager UI from DASH, you must have access to the following Netcool Agile Service Manager and DASH installation information:

DASH_PROFILE
The *DASH_Profile* variable describes the location of the application server profile that is used for Dashboard Application Services Hub. This location is in the `/profile/` subdirectory of the Jazz for Service Management home directory.

The default root user location is `/opt/IBM/JazzSM/profile`

The default non-root user location is `/home/<nonrootuser_name>/IBM/JazzSM/profile`

DASH_HOME
The *DASH_HOME* variable describes the (configurable) location where Dashboard Application Services Hub is installed.

The default root user location is `/opt/IBM/JazzSM/ui`

The default non-root user location is `/home/<nonrootuser_name>/IBM/JazzSM/ui`

NCHOME
The Netcool home location.

Usually found at `/opt/IBM/netcool`

Note: For more information on DASH and WAS environment variables, see the following topic in the Jazz for Service Management Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSEKCU_1.1.2.1/com.ibm.psc.doc/ref/psc_r_pathnames.html#psc_r_pathnames__tip_root

About this task

To uninstall the Netcool Agile Service Manager user interface from a DASH installation, you run the uninstall scripts included in the installation bundle. When uninstalling Netcool Agile Service Manager, you remove both the core services and the user interface. The recommended sequence of removal is to uninstall the UI first, and then remove the core.

The Netcool Agile Service Manager UI install bundle contains the install scripts and the `INASM.war` file.

Remember: You only use this uninstall process if you have **not** used the IBM Installation Manager for the UI installation. To uninstall versions of the UI that were installed using the IBM Installation Manager, use "Uninstalling the Netcool Agile Service Manager UI using the Installation Manager" on page 35.

When uninstalling, you remove the Netcool Agile Service Manager UI from an existing DASH installation.

Important: Ensure that you are logged in as the same user who installed DASH.

Procedure

Uninstall the Netcool Agile Service Manager UI

1. In a BASH command shell, open the INASM UI install bundle directory, and run `./uninstall.sh`
2. Type `Y` to confirm you are using the DASH installation user.
3. Provide the following information:

Option	Description
NCHOME	The default is `/opt/IBM/netcool`
DASH_PROFILE	The default is `/opt/IBM/JazzSM/profile`
DASH_HOME	The default is `/opt/IBM/JazzSM/ui`

   The uninstall process runs, uninstalling the Netcool Agile Service Manager UI from DASH.
4. Optional: Remove the INASM home directory in NCHOME. Enter `Y` at the prompt.

Restart DASH to allow the **uninstallation** to take effect.

5. To stop the DASH server, run `<DASH_PROFILE>/bin/stopServer.sh server1`
6. Once stopped, start the DASH server: `<DASH_PROFILE>/bin/startServer.sh server1`

Results

The Netcool Agile Service Manager UI has been removed from DASH.

What to do next

After uninstalling the UI, you remove the core services.

Note: You install and uninstall the latest version of the Netcool Agile Service Manager using the IBM Installation Manager.

Uninstalling the Netcool Agile Service Manager UI using the Installation Manager

To uninstall the Netcool Agile Service Manager user interface from a DASH installation, you use IBM Installation Manager. You only use this process to uninstall versions of the Netcool Agile Service Manager UI that were also installed with IBM Installation Manager.

Before you begin

When uninstalling Netcool Agile Service Manager, you remove both the core services and the user interface. The recommended sequence of removal is to uninstall the UI first using the IBM Installation Manager, and then remove the core.

About this task

Remember: If you have an installation of Netcool Agile Service Manager that was installed using the legacy, pre-IBM Installation Manager process, you must uninstall that version using the legacy uninstall process. See the following topic: "[BETA legacy] Uninstalling the Netcool Agile Service Manager UI" on page 33 Use IBM Installation Manager to remove Netcool Agile Service Manager UI.

Important: Ensure that you are logged in as the same user who installed DASH.

Procedure

1. Change to the `/eclipse` subdirectory of the Installation Manager installation directory.
2. Use the following command to start the Installation Manager wizard:
   `./IBMIM`
3. On the main Installation Manager window, click **Uninstall**.
4. Select **IBM Netcool Agile Service Manager**, then click **Next**. The installed package groups are displayed.
5. Under IBM Netcool GUI Components, select **IBM Netcool Agile Service Manager**, then click **Uninstall**. The user interface is uninstalled.
6. Click **Finish** to exit the Installation Manager.
7. After using the Installation Manager to uninstall Netcool Agile Service Manager, you must restart DASH to ensure that it recognizes the removal of Netcool Agile Service Manager.

Results

The Installation Manager removes the files and directories that it installed, leaving behind the application configuration file and log files.

What to do next

After uninstalling the UI, you remove the core services.

Uninstalling the Netcool Agile Service Manager core services

To uninstall the Netcool Agile Service Manager core, you remove `nasm-common`. Due to package dependencies, this will stop and remove **all** of the Netcool Agile Service Manager core Docker containers and images, and then remove the installation files from your server.

Before you begin

To uninstall Netcool Agile Service Manager, you remove both the core services and the user interface. Uninstall the UI first, and then remove the core as described in this procedure.

About this task

The following procedure sequentially stops and removes all Netcool Agile Service Manager Docker packages, before removing the images and then the installation packages from the server. Any other Docker components are not affected, and the Docker service is not stopped.

Note: Although you can remove individual packages, this is the recommended uninstall procedure for Netcool Agile Service Manager core components.

Tip: Use the following command to see a list of the installed docker containers:

```
docker ps -a
```

Procedure

Use the following command to stop the server, remove the installation images from Docker, and then remove the rpm packages from the server.

```
sudo yum remove nasm-common
```

Results

The Netcool Agile Service Manager core services have been removed.

Installing and configuring on IBM Cloud Private

To install Netcool Agile Service Manager on IBM Cloud Private, you follow the NOI documentation installation steps, together with the steps described here.

Installation process overview: To install NOI with Agile Service Manager, you ensure that you have adequate system resources to be able to deploy the system, prepare the required storage cluster, before downloading the Agile Service Manager software and importing it into IBM Cloud Private. Then, you prepare the installation configuration file, and then run the installation.

Remember: You cannot upgrade from ibm-netcool-asm-prod version 1.0.0 to 2.0.0, or 2.0.0 to 3.0.0, using the Helm upgrade command or via the ICP console. To upgrade from version 1.0.0 to version 2.0.0, or from version 2.0.0 to version 3.0.0, you need to remove the existing deployment and then reinstall, as documented in

the Release Notes. If you already have ibm-netcool-asm-prod version 3, you can upgrade to a later version using the command line and ICP console, as described here.

Related information:

https://www-03preprod.ibm.com/support/knowledgecenter/SS9LQB_1.1.5/ReleaseNotes/asm_rn_11_5.html

Installing Agile Service Manager on ICP

Before installing Agile Service Manager, you follow the NOI installation steps in the Netcool Operations Insight documentation. This topic describes the installation steps specific to Agile Service Manager only.

Before you begin

Important: Ensure you have the latest version of the IBM Netcool Operations Insight documentation.

Encrypting Storage: As an optional security enhancement, you can deploy virtual machines with encrypted storage volumes on IBM Cloud Private.

IBM Cloud Private documentation
: Encrypting vSphere volumes

  Encrypting volumes by using dm-crypt

Encrypting Agile Service Manager data
: You mount `/opt/ibm/netcool/asm` from an encrypted volume, and then install Agile Service Manager on that mounted volume.

  Alternatively, you can backup an existing installation, and then move (restore) it onto the encrypted volume.

Encryption use
: A typical encrypted system would be configured to be mounted (and then unlock access to the encrypted data) when booting. Without a pass-phrase or key the server with the encrypted volume will not boot up.

  Instead of storing this pass-phrase or key together with the encrypted volume, you should provide it while the server is booting, either on a removable device, or by entering it manually.

Remember:

The NOI installation procedures in the Netcool Operations Insight documentation include the following steps specific to Agile Service Manager for ICP:

Prepare your ICP system
: This procedure is described in the 'Preparing for installation on IBM Cloud Private' topic of the NOI documentation.

  While completing the NOI step, you will have prepared a fully functioning cluster on IBM Cloud Private.

  Ensure that your system satisfies the minimum Agile Service Manager requirements:

  Kubernetes
  : Version 1.11.0+

Worker nodes
A minimum of three worker nodes with adequate storage

Number of CPUs (per worker node)
48

Memory (per worker node)
64 GB

Storage (per worker node)
300 GB

Processor
amd64

IBM Cloud Private version
3.2.0 or later

Download the Agile Service Manager package
This procedure is described in the 'Downloading for installation on IBM Cloud Private' topic of the NOI documentation.

As part of completing the NOI step, you will have downloaded the Agile Service Manager package. The download file contains all images that you require to run the software, as well as the Helm charts used to install the images.

Ensure you have obtained the latest package.

Load the archive into IBM Cloud Private
You must have the cluster administrator role to load the archive.

This procedure is described in the 'Loading the archive into IBM Cloud Private' topic of the NOI documentation.

While completing the NOI step, you will have imported the Agile Service Manager package into ICP. The download file contains all images that you require to run the software, as well as the Helm charts used to install the images.

Upgrade Note:

After Agile Service Manager has been uninstalled, orphaned job objects and related pods may remain on the system. Remove these as in the following example:

```
kubectl delete job -l release=myReleaseName --cascade
```

About this task

The following procedure describes how to perform prerequisite tasks to prepare the installation environment, how to edit the installation configuration file, and how to then install Agile Service Manager from the command line.

Pre-requisite steps: You run the following three scripts to complete the prerequisite steps before installation.

createSecurityClusterPrereqs.sh
Creates the PodSecurityPolicy and ClusterRole for all releases of this chart.

createStorageVolumes.sh
Creates the required storage volumes for a single deployment of the chart.

createSecurityNamespacePrereqs.sh
Creates the ClusterRoleBinding for the namespace.

Procedure

Prepare the Agile Service Manager environment

Note: You **must** run the prerequisite scripts as cluster administrator.

1. Extract the scripts from the `pak_extensions/pre-install/clusterAdministration` directory. Use the following command:

   ```
   $ tar xvf ibm-netcool-asm-v1.1.5-x86_64.tar.gz pak_extensions/
   ```

2. To create the custom pod security policy, run the following command:

   ```
   createSecurityClusterPrereqs.sh
   ```

   The script creates the pod security policy 'ibm-netcool-asm-prod-psp'.

3. Create the required storage volumes for a single deployment of the chart.
   a. Add all required configuration data, such as worker node URLs, disk locations and capacity, to the storageConfig.env file For example:

      ```
      WORKER1=172.99.0.1
      WORKER2=172.99.0.2
      WORKER3=172.99.0.3
      FS_ROOT=/opt/ibm/netcool
      # Volume capacity in Gi
      CAPACITY_CASSANDRA=50
      CAPACITY_KAFKA=15
      CAPACITY_ELASTICSEARCH=75
      CAPACITY_ZOOKEEPER=5
      # (Optional) File Observer Settings
      FILE_OBSERVER_DATA_CAPACITY=5
      FILE_OBSERVER_DATA_NODE=${WORKER1}
      ```

   b. Create the storage volume, specifying the namespace and release name for the install.

      ```
      bash createStorageVolumes.sh myNamespace myReleaseName
      ```

      The script will print a list of the directories that you must create on each of the worker nodes. For example:

      ```
      WARN: You need to manually create these paths on each node before the
      volumes can be used:
      172.16.185.83 /opt/ibm/netcool/asm/data/cassandra-0
      172.16.185.83 /opt/ibm/netcool/asm/data/elasticsearch-2
      172.16.185.83 /opt/ibm/netcool/asm/data/kafka-1
      172.16.185.83 /opt/ibm/netcool/asm/data/zookeeper-1
      172.16.193.151 /opt/ibm/netcool/asm/data/cassandra-1
      172.16.193.151 /opt/ibm/netcool/asm/data/elasticsearch-0
      172.16.193.151 /opt/ibm/netcool/asm/data/kafka-2
      172.16.193.151 /opt/ibm/netcool/asm/data/zookeeper-2
      172.16.195.206 /opt/ibm/netcool/asm/data/cassandra-2
      172.16.195.206 /opt/ibm/netcool/asm/data/elasticsearch-1
      172.16.195.206 /opt/ibm/netcool/asm/data/kafka-0
      172.16.195.206 /opt/ibm/netcool/asm/data/zookeeper-0
      ```

   c. Create the directories on each of the worker nodes.

4. To assigns the pod security policy to the namespace, create the ClusterRoleBinding.

   ```
   bash createSecurityNamespacePrereqs.sh myNamespace
   ```

5. **On each worker node**, set the Cassandra and Elasticsearch **vm.max_map_count** kernel parameter to a value of at least 1048575.
   a. Use `sysctl` to set the parameter with immediate effect.

      ```
      sysctl -w vm.max_map_count=1048575
      ```

b. Update the `/etc/sysctl.conf` configuration file to ensure that the change remains in effect after a restart by adding the following line.

```
vm.max_map_count=1048575
```

Deploy Agile Service Manager

6. Add the internal ICP Helm repository to the Helm configuration. This process is described in the following topic of the IBM Cloud Private Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.2.0/app_center/add_int_helm_repo_to_cli.html
7. Deploy the Agile Service Manager installation from the command line. The following example command installs the Agile Service Manager chart (ibm-netcool-asm-prod-2.0.0.tgz) into a namespace called `netcool` with a release name of `asm`:

```
$ helm install --name asm --namespace netcool
ibm-netcool-asm-prod-3.0.0.tgz --set license=accept --tls
```

What to do next

You login to the Agile Service Manager ICP installation using a URL of the following format (example):

```
https://netcool.noi.icp-master.<your_host>/ibm/console
```

Where *noi* is the Netcool Operations Insight Helm release name. Use the following command to retrieve the DASH URL:

```
helm status NOI helm release name --tls
```

Tip: If you want to encrypt all intra-cluster communication using IPSec, you can find more information on the IBM Cloud Private Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.2.0/installing/ipsec_mesh.html

Note:

If you are installing an updated version of Agile Service Manager on ICP, and your services fail to bind to the existing provisioned storage, see the Services not binding to storage troubleshooting topic.

Related information:

- Kubernetes Documentation
- IBM Cloud Private role-based access control

Configuring DASH user roles

You configure DASH user roles so that users can use the Netcool Agile Service Manager UI. This task is the same for both on-prem and ICP deployments of Agile Service Manager.

About this task

You can assign the following DASH user roles to users:

inasm_operator
: A user with the inasm_operator role can access the Netcool Agile Service Manager UI, and use it to search for and visualize the resources in the Netcool Agile Service Manager core application.

inasm_editor
The same as for inasm_operator.

In addition, a user with the inasm_editor role can add comments to resources from the Topology Viewer Context (right-click) menu. (A user with the inasm_operator role can view comments, but not add new ones.)

inasm_admin
The same as for inasm_editor.

In addition, a user with the inasm_admin role has access to a number of administrator tools, where they can define custom UI elements for the Topology Viewer. See the "Customizing UI elements" on page 191 topic for more information.

To configure DASH user roles, you must log into DASH with admin user credentials.

Tip: You can also assign roles to a user indirectly by assigning them to a group of which the user is a member.

Procedure

1. As the admin user, log into your DASH web application.

   For on-prem
   If you have used the default root location of `/ibm/console`, use the following logon URL:

   `https://<DASH-HOST>:<DASH-PORT>/ibm/console/logon.jsp`

   For ICP

   You login to the Agile Service Manager ICP installation using a URL of the following format (example):

   `https://netcool.noi.icp-master.<your_host>/ibm/console`

   Where *noi* is the Netcool Operations Insight Helm release name. Use the following command to retrieve the DASH URL:

   `helm status NOI helm release name --tls`
2. Select **Console Settings** from the DASH menu.
3. Select **User Roles** from the Console Settings menu (under the Roles heading).
4. Select the **User Roles** tab, and then click **Search**. Known users are displayed in the Search results table.
5. **For each user** requiring access to the Netcool Agile Service Manager UI, perform the following actions:
   a. Click the required user ID in the Search results table. All roles that are available to the selected user are displayed in the Available Roles table.
   b. Select the required roles, as appropriate.
   c. Click **Save**.

Results

Once you have saved your changes, the user role changes take effect. All users with their newly assigned roles are now able to log into DASH and access the Netcool Agile Service Manager UI. From there, users can search and visualize resources from the Netcool Agile Service Manager topology service.

Remember: You can also assign roles to a user by assigning them to a group to which the user belongs.

Uninstalling Agile Service Manager

To uninstall Agile Service Manager on IBM Cloud Private, you uninstall Agile Service Manager, and then run a number of post-delete cleanup scripts.

Before you begin

You can identify the chart to uninstall by using the `helm list --tls` command. The system will return information identifying the deployed charts, as well as other information, such as a chart's namespace. The following examples use *myReleaseName* and *myNamespace* to indicate these.

About this task

The post-delete scripts are located in the `pak_extensions/post-delete` directory.

Procedure

1. Run the following Helm command to uninstall Agile Service Manager on IBM Cloud Private.

   ```
   helm delete myReleaseName --purge --tls
   ```

   Where *myReleaseName* is the name of the release that you specified when you installed Agile Service Manager on IBM Cloud Private.

   Note: The `--purge` option removes the release from the store and frees its name for later use.
2. As cluster or team administrator, delete the RoleBinding for the namespace. For example:

   ```
   bash deleteSecurityNamespacePrereqs.sh myNamespace
   ```
3. As cluster administrator, delete the PodSecurityPolicy and ClusterRole for all releases of this chart. For example:

   ```
   bash deleteSecurityClusterPrereqs.sh
   ```
4. As cluster administrator, remove the persistent storage volumes and claims for a release. For example:

   ```
   bash deleteStorageVolumes.sh myReleaseName
   ```
5. Clean up remaining cron job objects and their related pods.

   After Agile Service Manager has been uninstalled, orphaned job objects and related pods may remain on the system. Remove these as in the following example:

   ```
   kubectl delete job -l release=myReleaseName --cascade
   ```

Results

Agile Service Manager and remaining files have been removed.

If you are installing an updated version of Agile Service Manager on ICP, and your services fail to bind to the existing provisioned storage, see the Services not binding to storage troubleshooting topic.

ICP on OpenShift reference

Agile Service Manager Version 1.1.5 is supported on IBM Cloud Private Version 3.2.0, running on RedHat OpenShift Container Platform (OCP) Version 3.11. Use this topic as a reference when installing such a system.

Deployment of ICP on OpenShift

Assumptions:

IBM Cloud Private has been installed on Red Hat OpenShift. See the following ICP topic for more information: IBM Cloud Private with OpenShift

The following command line tools are available and configured for use:

- oc
- cloudctl
- docker
- helm

Storage

You can use local storage volumes, or configure external storage, as documented here:
"Configuring the Helm chart to use alternate storage (ICP on OpenShift)" on page 207

Load PPA package

Find the OCP docker registry

ICP normally includes an image registry. When ICP is deployed on OCP, you instead make use of the image registry provided by OCP. The registry can normally be found in the `default` namespace, although this is not guaranteed.

To find the docker registry:

```
# oc get pod --all-namespaces -l deploymentconfig=docker-registry
NAMESPACE   NAME                      READY     STATUS    RESTARTS   AGE
default     docker-registry-1-kxqm5   1/1       Running   2          21d
```

The information returned confirms that the registry is in the `default` namespace.

Check the service name

```
# oc get svc -n default
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                   AGE
docker-registry    ClusterIP   172.30.196.240   <none>        5000/TCP                  21d
kubernetes         ClusterIP   172.30.0.1       <none>        443/TCP,53/UDP,53/TCP     21d
registry-console   ClusterIP   172.30.117.199   <none>        9000/TCP                  21d
router             ClusterIP   172.30.68.171    <none>        80/TCP,443/TCP,1936/TCP   14d
```

The information returned confirms that the docker registry is accessible via `docker-registry.default.svc:5000`.

For other namespaces or service names the format is:
`{SERVICE_NAME}.{NAMESAPCE}.svc:5000`

Login docker registry

Using both the oc and docker command line tools, you can run the following command to authenticate with the docker registry:

```
# docker login -u $(oc whoami) -p $(oc whoami -t) docker-registry.default.svc:5000
Login Succeeded
```

cloudctl load-archive

You load the Agile Service Manager archive in much the same way as with normal ICP, except you must point to the OCP docker registry, and append the namespace in which you are planning to install. In the following example you use a namespace called `netcool`.

```
# cloudctl catalog load-archive --archive ibm-netcool-asm-v1.1.5-x86_64.tar.gz
--registry docker-registry.default.svc:5000/netcool
```

The output should be similar to the following sample:

```
Importing docker image(s)
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-ui-api:1.0.0-11502
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-cienablueplanet-observer:1.0.6.11514
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-vmwarensx-observer:1.0.16.11523
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-dns-observer:1.0.7.11510
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-bigfixinventory-observer:1.1.0.11518
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-merge-service:1.0.5.11509
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-docker-observer:1.0.10.11513
    Pushing image as: docker-registry.default.svc:5000/netcool/nasm-aws-observer:1.0.0.11516
OK

Uploading Helm chart(s)
  Processing chart: charts/ibm-netcool-asm-prod-3.0.0.tgz
  Updating chart values.yaml
  Uploading chart
Loaded Helm chart
OK
```

Install Agile Service Manager from the ICP console

Get routes

Before accessing the ICP console (UI), you need to check the `routes` in the `kube-system` namespace.

```
# oc get routes -n kube-system
NAME         HOST/PORT           PATH        SERVICES               PORT   TERMINATION          WILDCARD
icp-console  icp-console.apps.bert.acme.co.uk  icp-management-ingress  <all>  passthrough/Redirect None
icp-proxy    icp-proxy.apps.bert.acme.co.uk    nginx-ingress          <all>  passthrough/Redirect None
```

Install Agile Service Manager

Now you can access, configure and install the Agile Service Manager chart from the ICP console.

Note: You may see various warnings related to pod security policies. As these are not applicable to OpenShift, you can ignore the warnings. This is a known ICP defect.

Install Agile Service Manager from the command line

Find the repository URL

Before installing Agile Service Manager from the command line, you must find the repository URL using the following command:

```
# cloudctl catalog repos
Name                  URL                                                       Local
ibm-charts            https://raw.githubusercontent.com/IBM/charts/master/repo/stable/      false
local-charts          https://icp-console.apps.bert.acme.co.uk:443/helm-repo/charts         true
mgmt-charts           https://icp-console.apps.bert.acme.co.uk:443/mgmt-repo/charts         true
ibm-charts-public     https://registry.bluemix.net/helm/ibm/                                false
ibm-community-charts  https://raw.githubusercontent.com/IBM/charts/master/repo/community/   false
ibm-entitled-charts   https://raw.githubusercontent.com/IBM/charts/master/repo/entitled/    false
```

In this example, the URL is for repository `local-charts`:
`https://icp-console.apps.bert.acme.co.uk:443/helm-repo/charts`

Add URL to repository

After finding the URL, you add it to the repository:

```
# helm repo add icp-repo https://icp-console.apps.bert.acme.co.uk:443/helm-repo/charts --ca-file .helm/ca.pem
"icp-repo" has been added to your repositories
```

Install Agile Service Manager

Install Agile Service Manager using the helm command line:

```
# helm search icp-repo/ibm-netcool-asm-prod
NAME                        CHART   VERSION   APP     VERSION   DESCRIPTION
icp-repo/ibm-netcool-asm-prod    3.0.0            1.1.5           IBM® Netcool® Agile Service Manager
```

Ingress

ICP uses nginx as an ingress controller. OCP uses ha-proxy, which can cause issues when both are deployed. Both act as a http proxy into the cluster. Both want to listen on http/https ports, so getting both installed requires that the nginx-controller is installed and configured to listen on different ports than normal.

ICP creates a number of routes to the console and to the nginx proxy.

OCP can create routes automatically from Ingress objects, but you should be aware of the following considerations:

- Ingress should work exactly the same as is does today, as long as you use the ICP provided nginx ingress point.
- If you have not set a hostname in the ingress rule, you can reach Agile Service Manager via the icp-proxy route.
- If you do set a hostname, you can't access Agile Service Manager via the default icp-proxy route.
- If you do set a hostname, if will work if you ingress via nginx, rather than the OCP router.
- To access via nginx you need to use alternate ports. The ICP nginx ingress does **not** run on the usual ports, which are used by the OCP router.
- The alternate ports are configured when you install ICP:

```
# egrep '^ingress_http.?_port' cluster/config.yaml
 ingress_http_port: 3080
 ingress_https_port: 3443
```

Known issues: Launch links from the ICP UI continue to suffer the following issues (as for previous releases):

- They do not work from the deployment page.
- They do not work with a non-default ingress hostname.
- They only work with no hostname set (the Agile Service Manager default).

Table 9. Topology service access. Using a cluster proxy node name of **bert.acme.co.uk**, the following table shows access to the topology service.

global.ingress. domain	global.ingress. tlsSecret	ocp-router	nginx-ingress
		https://icp-proxy. apps.bert.acme.co.uk /1.0/topology	https://icp-proxy. apps.bert.acme.co.uk: 3443/1.0/topology
asm.bert.acme.co.uk			http://asm.bert.acme.co.uk: 3080/1.0/topology
asm.bert.acme.co.uk	asm-tls-secret		https:// asm.bert.acme.co.uk: 3443/1.0/topology

Installing and configuring a hybrid ICP / on-prem system

You can install Netcool Agile Service Manager with an ICP backend, and access it via an Agile Service Manager UI deployed in DASH (with NOI).

Assumption: The hybrid system deployment described here is intended specifically for managing event data extracted from Netcool/OMNIbus.

Configuring a hybrid installation

To create your hybrid system, you first connect the Agile Service Manager on-prem UI to the Agile Service Manager ICP UI-API service. You then configure the connection of the hybrid system with the existing on-prem NOI Netcool/OMNIbus deployment.

Procedure

Connecting the on-prem UI to the ICP UI-API service

1. Export the ICP client SSL certificate.

   ```
   echo -n | openssl s_client -connect <ICP HOST>:443 | sed -ne '
   /-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/icp.cert
   ```

2. Import the certificate into the OnPrem truststore. See the `application.yml` file for passwords.

   ```
   keytool -importcert -keystore /opt/IBM/gui/inasm/security/truststore.p12
   -trustcacerts -keypass <PASSWORD> -storepass <PASSWORD> -file /tmp/icp.cert
   -storetype PKCS12
   ```

3. Update the following `opt/IBM/gui/inasm/etc/application.yml` file properties:
   a. Update the **`proxyServiceHost`** parameter with the ICP master node hostname.
   b. Update the **`namespace`** parameter with the Agile Service Manager on ICP value (for example `netcool`).
   c. Remove the deprecated **`proxyServiceUsername`** and **`proxyServicePassword`** configuration parameters.
4. Restart DASH.

Connecting the Agile Service Manager ICP installation to the existing on-prem NOI Netcool/OMNIbus system

5. The integration requires a column called **`LastOccurrenceUSec`** in the alerts.status and alerts.problem_events tables of the external ObjectServer. If the column does not exist, you apply the provided `asmclear.sql` script to the external ObjectServer (or servers).

   Note: The `asmclear.sql` script alters alerts.status and alerts.problem_events by inserting the LastOccurrenceUSec column, and also alters the generic_clear trigger. Proceed with caution if you have customized any of these on your system.

   - Run the `asmclear.sql` script on every ObjectServer on all layers of a multi-tiered Netcool/OMNIbus architecture.
6. Ensure the Netcool/OMNIbus probe for Message Bus and the XML Gateway for Event Observer can reach the external ObjectServer installation.
7. Add a secret for the O/S root account password, for example:

   ```
   kubectl create secret generic external-noi-omni-secret
   --from-literal=OMNIBUS_ROOT_PASSWORD=XXXXXXXXX --validate=true
   ```

8. Create a modified values file for the helm chart by appending the following data into the appropriate yaml file (for example, the `asm-config-noi-onprem.yaml` file).

```
# where to find noi
noi:
  releaseName: external-noi
  credentialsSecret:     external-noi-omni-secret
  omnibus:
    primary:
       name: <Primary OnPrem Object Server Host>
       port: 4100
    backup:
      name: <Backup OnPrem Object Server Host>
      port: 4100
```

9. Delete any 'job' pods:

```
kubectl delete job --all
```

10. Upgrade the install of Agile Service Manager, using the updated values file, for example:

```
helm upgrade <release_name> <helm_chart_repository>/ibm-netcool-asm-prod
--tls --values=asm-config-noi-onprem.yaml
```

Chapter 4. Running Observer jobs

Agile Service Manager is available with a large number of observers, and can be deployed as on-prem or IBM Cloud Private versions. Not all observers are available on IBM Cloud Private.

observer

An observer is a service that extracts resource information and inserts it into the Agile Service Manager database.

Agile Service Manager includes a configuration UI to help you configure and run observer jobs.

Observer job names

The characters that you can use when defining the Unique IDs (**unique_id**) of Agile Service Manager observer jobs have been restricted to the following:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789-
._~:#[]@!$&'()*+,;=
```

If you have used any characters other than these (such as '/'), you must recreate the job definition.

Defining observer security

All observer jobs require password encryption, and in addition some observer jobs require authentication credentials. This topic describes such configuration tasks for both ICP and on-prem versions of Agile Service Manager.

For the IBM Cloud Private version of Agile Service Manager, observer jobs are defined and run using Swagger. For information on how to customize and deploy observers on ICP, see the included Swagger documentation.

Remember: Swagger links to specific observers are in “Swagger reference” on page 311, and more information on specific observers is located in the subtopics under “Observer reference” on page 100.

Configuring password encryption, authentication certificates and keystores

All observer jobs require password encryption, and in addition some observers require authentication credentials. This topic describes such configuration tasks for the ICP and on-prem versions of Agile Service Manager, and also describes how to post an observer job using Swagger (or cURL).

About this task

The Kubernetes Observer requires a certificate for authentication, and the VMware NSX, VMware vCenter, Cisco ACI, Bigfix Inventory, Zabbix, and Ciena Blue Planet observers require both certificates and keystores.

You must encrypt **all** observer jobs.

The following steps are described:

- Encrypt the passwords for all observer load and listen jobs
- Obtain an authentication certificate
- Store that certificate as a secret
- Post an observer job

Procedure

Encrypt the passwords for all observer load and listen jobs

1. The jobs for all observers require the password in the configuration file to be encrypted. To encrypt the password, use the commands in the following example:

   ```
   kubectl exec -ti asm-topology-pods- -- java -jar /opt/ibm/topology-service/topology-service.jar
   encrypt_password --password 'password'
   ```

   Where the value of *asm-topology-pods* can be obtained using the following command:

   ```
   kubectl get pods | grep topology
   myasm-topology-9f98c8448-ckxpp
   ```

   The encryption utility will return an encrypted password

To acquire an SSL certificate and build the SSL truststore (on-prem)

2. Use the following **Cisco ACI Observer example** to acquire an SSL certificate. In the following example, you use OpenSSL to connect to Cisco APIC over port 443, and extract a SSL Certificate from Cisco APIC to a *<certificate_file_name>*.crt file.

   ```
   echo -n | openssl s_client -connect {Cisco APIC IpAddress}:443 | sed -ne
   '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
   ```

3. Use the following example to import a certificate file into a keystore and encrypt the keystore. Use the following Java keytool command to import the Cisco APIC certificate file into a keystore and encrypt the keystore with a given password.

   ```
   keytool -import -v -trustcacerts -alias {Cisco APIC Hostname}
   -file {certificate_file_name}.crt -keystore {keystore file name}
   -storepass {your plain text password to encrypt keystore}
   ```

   Tip: You will need the following encryption information when editing ciscoaci_observer_common.sh

Table 10. Encryption parameters required for ciscoaci_observer_common.sh

keystore parameter	ciscoaci_observer_common.sh parameter
keystore password	password_ssl_truststore_file
keystore file name	ssl_truststore_file

4. Copy the keystore file ({keystore file name}) to the $ASM_HOME/security directory to complete the SSL setup.

Managing authentication certificates and storing them as secrets (ICP)

5. Obtain the authentication certificate using OpenSSL.

   ```
   echo -n | openssl s_client -connect {ipAddress}:{port} | sed -ne '/-BEGIN CERTIFICATE-/,/
   -END CERTIFICATE-/p' | base64 -w 0 > target_system.crt
   ```

   Where target_system.crt contains the encoded certificate, and {ipAddress} could be the IP address of any of the following target systems:

   - VMware vCenter
   - VMware NSX

- Cisco ACI
- Kubernetes master node
- Bigfix Inventory
- Zabbix
- Ciena Blue Planet

Example `target_system.crt`:

```
[root@localhost ~]# cat target_system.crt
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN3RENDQWFnQ0NRRGRuMENqU3BXZXhUQU5CZ2txaGtpRzl3
MEJBUVVGQURBaU1RMHdDd1lEVlFRERERBUkIKVUVsRE1SRXdEd1lKS29aSWh2Y05BUWtCRmdKVlV6QWVGdzB4
TmpBeE1qRXdOekV5TWpWYUZ3MHlOakF4TVRndwpOekV5TWpWYU1DSXhEVEFMQmdOVkJBTU1CRUZRU1VNeEVUQVBCZ2txa
GtpRzl3MEJDUUVXQWxWVE1JSUJJakFOCkJna3Foa2lHOXcwQkFRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQW1Ob0dxd
FI0RlFPWkdoUWFtand1YmxRYjRobU0KTzJwOGtjbGUwL2NuUno3cSttWGYzWlRQYTZsWEk2MG9BbmVPSGowZEVa
MkhwRWFFblBUbWJmWUF6Y0ZQdjNVWApMWjM3VWVoMDZXTjMxS29tSSs2czJtSk1IWWM0MW44M1RiUU5uWUNjYjZjdlZLc
WV5NVhhaFBtdkZDbDBtM3Y3Cisxa1lFMFRNVlBnTk56R0ZSUXU1RVlGc3FZWHZGbFZhZ0lUalF6cks3YnE0Rk
JiMW1kVjFsYnVOMWhISzd2SFEKS3ZUNHBGbGx1NTRHU0JhZ2RSbUdad0dta0tNZHRGUkEvc3pBWEMrejQ0cHN3T05yd
TJnbDR3bG5MZTVvM2NWZwpFQUx1THM4UDgrOUx0eFN3YWJvb0VMcHRjb3pKdEpUb2E4QS9zZXRaSi81RUJQNmhj
Nk1yUWxHQktRSURBUUFCCk1BMEdDU3FHU0liM0RRRUJCUVVBQTRJQkFRQkJuZzlJK2pBdjhNUjBYemM1SUUxd
TBkK0JweW9OZGVRbkl4T2sKZWFsNzNUbmkzWmh4QUQzdlQzenZNSElSUEc0d3lxMWJqQ05LY3BZOGVCbVJuVzhOSnl
NdG9vcU9hNlJMWGNPTAoyeVZublVna092THRPVjM5eFNFQ1BOMzV4YXJJdGYydE9NZWJRWlc1ZC9Hc1lPZUFLTl
NrTlQwRmtreDE0UzJFClpBVi9IUUVHaVpUR0tQNkx1czYzLzJiTEJVNHdGUjg3bjNkdFJFVUp5eGQ4
ZDJDTFA4MkE2UTNOT2lOZEdkam0KSnFQZXNEaWxXWE5Gd09xUk1XOWFGWTVUSUtOL25PQzhqczI0cVFm
ZTJZcllnZ242N0crLytBQy9kV21JSVQ2dgpBWTVMejhwOWQwSzZUaGxLeVpNZkdYVkNnMFlvTms1ajQ4ckJlZ2J5c
FhTM1J2SnIKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=

[root@locahost ~]#
```

Tip: To get the ipaddress and port for each respective observer, see "Swagger reference" on page 311 or the observer subtopics under "Observer reference" on page 100

6. Store that certificate as a secret. Each installed Agile Service Manager release has a single special secrets file. Data added to that is made available to the appropriate observer containers.

   ```
   $ kubectl edit secret asm-custom-secrets
   ```

   Paste in the encoded certificate generated in the previous step.

   a. Find the correct secrets file using the following command:

      ```
      $ kubectl get secrets -l app=ibm-netcool-asm-prod
      NAME TYPE DATA AGE
      asm-custom-secrets Opaque 2 29d
      ```

   b. Edit the appropriate file for your release.

      ```
      $ kubectl edit secret asm-custom-secrets
      ```

   c. Add a name and value pair to the data section. The value is the certificate generated earlier. The name is what you enter as the certificate file name to run the observer job.

      ```
      data:
      {name}:{value}
      ```

      Example of expected content in the secret file after adding vcenter.crt is as follows (where the data section is between the 'apiVersion' and 'kind' sections).

      Note: This VMware vCenter Observer example registers the vcenter.crt SSL certificate in ICP Secret, and vcenter.crt is the job parameter value for the VMware vCenter Observer. Define a new **{name}** parameter in the same file for other observers that require SSL certificates.

      ```
      apiVersion: v1
      data:
      vcenter.crt:LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN3RENDQWFnQ0NRRGRuMENqU3BXZXhUQU5CZ2txaGtpRzl3
      MEJBUVVGQURBaU1RMHdDd1lEVlFRERERBUkIKVUVsRE1SRXdEd1lKS29aSWh2Y05BUWtCRmdKVlV6QWVGdzB4
      TmpBeE1qRXdOekV5TWpWYUZ3MHlOakF4TVRndwpOekV5TWpWYU1DSXhEVEFMQmdOVkJBTU1CRUZRU1VNeEVUQVBCZ2txa
      GtpRzl3MEJDUUVXQWxWVE1JSUJJakFOCkJna3Foa2lHOXcwQkFRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQW1Ob0dxd
      ```

```
FI0RlFPWkdoUWFtand1YmxRYjRobU0KTzJwOGtjbGUwL2NuUno3cSttWGYzWlRQYTZsWEk2MG9BbmVPSGowZEVa
MkhwRWFFblBUbWJmWUF6Y0ZQdjNVWApMWjM3VWVoMDZXTjMxS29tSSs2czJtSk1IWWM0MW44M1RiUU5uWUNjYjZjdlZLc
WV5NVhhaFBtdkZDbDBtM3Y3Cisxa1lFMFRNVlBnTk56R0ZSUXU1RVlGc3FZWHZGbFZhZ0lUalF6cks3YnE0Rk
JiMW1kVjFsYnVOMWhISzd2SFEKS3ZUNHBGbGx1NTRHU0JhZ2RSbUdad0dta0tNZHRGUkEvc3pBWEMrejQ0cHN3T05yd
TJnbDR3bG5MZTVvM2NWZwpFQUx1THM4UDgrOUx0eFN3YWJvb0VMcHRjb3pKdEpUb2E4QS9zZXRaSi81RUJQNmhj
Nk1yUWxHQktRSURBUUFCCk1BMEdDU3FHU0liM0RRRUJCUVVBQTRJQkFRQkJuZzlJK2pBdjhNUjBYemM1SUUxd
TBkK0JweW9OZGVRbkl4T2sKZWFsNzNUbmkzWmh4QUQzdlQzenZNSElSUEc0d3lxMWJqQ05LY3BZOGVCbVJuVzhOSnl
NdG9vcU9hNlJMWGNPTAoyeVZublVna092THRPVjM5eFNFQ1BOMzV4YXJJdGYydE9NZWJRWlc1ZC9Hc1lPZUFLTl
NrTlQwRmtreDE0UzJFClpBVi9IUUVHaVpUR0tQNkx1czYzLzJiTEJVNHdGUjg3bjNkdFJFVUp5eGQ4
ZDJDTFA4MkE2UTNOT2lOZEdkam0KSnFQZXNEaWxXWE5Gd09xUk1XOWFGWTVUSUtOL25PQzhqczI0cVFm
ZTJZcllnZ242N0crLytBQy9kV21JSVQ2dgpBWTVMejhwOWQwSzZUaGxLeVpNZkdYVkNnMFlvTms1ajQ4ckJlZ2J5c
FhTM1J2SnIKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
kind:Secret
```

If the edit is successful, the following message will be displayed:

```
secret "asm-custom-secrets" edited
```

7. In the ICP GUI, you can view the configured secret under **Menu** > **Configuration** > **Secrets** option, where the **Name** is 'asm-custom-secrets'. Within asm-custom-secrets, all data configured earlier is displayed.

Posting a job

8. Post the job via the Swagger UI or cURL.

   Note: For the VMware NSX, VMware vCenter and Cisco ACI observers the default value for the `password_ssl_truststore_file` property is **password** and has to be encrypted.
   Example cURL command:

```
curl --location --insecure  --header 'Content-Type: application/json' --header
'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' -d '{
  "unique_id": "my job",
  "type": "query",
  "parameters": {
    "data_center": "LondonDC1",
    "vcenter_username": "admin",
    "vcenter_password": "RW+w==",
    "vcenter_api_url": "https://localhost/rest",
    "vcenter_certificate": "vcenter.crt",
    "ssl_truststore_file": "localhost.jks",
    "password_ssl_truststore_file": "IxcQ9w==",
    "connect_read_timeout_ms": 5000
  }
}' 'https://<master-ip address>/1.0/vmvcenter-observer/jobs/restapi'
```

   Note: When using cURL, you may need to add `--location` so that it will follow redirects, and `--insecure` as the proxy server is using HTTPS.

What to do next

For a repeating job, you can wrap the cURL in a script and use a normal cron job.

Defining observer jobs using the Observer Configuration UI

You configure observer jobs using the Observer Configuration UI, which you access through DASH. To schedule jobs and configure truststore certificates, you use the information in the Reference sections that describe manual observer job configuration.

1. Using a compatible browser, open DASH using the DASH URL. For example:

   ```
   https://<DASH HOST>:<DASH PORT>/ibm/console/
   ```

2. Login using your user credentials.
3. In DASH, open the **Administration** menu.

4. Under the Agile Service Management heading, click **Observer jobs** to display the Observer Configuration UI. From here, you can search for an existing job, or open either the **Existing jobs** or **Add a new job** expandable sections.

Existing jobs

The Existing jobs panel displays all jobs as tiles.

Each job state is indicated (Pending, Running, Stopping, Stopped, Finished)

From here, you can run a job or switch it On or Off, depending on the job type.

You can also use the **List of options** drop-down to either **View & Edit**, or **Delete** a job.

Add a new job

The Add a new job panel displays all jobs that can be configured in tile format.

Click the **Configure** button under a specific observer to open its job configuration window.

The Observer job configuration UI lists each job parameter that you have to configure.

Tip:

If an observer has been stopped or removed, you will be unable to run existing jobs, or add new jobs. Stopped or removed observers and jobs that are listed in the Observer Configuration UI will be disabled (grayed out) or removed in progressive (housekeeping) steps. If you are reinstalling or reactivating an observer, the jobs and the observer will again become available.

1. **Up to 5 minutes** after removal, observers and jobs still appear as normal until the housekeeping process runs, but cannot be used.
2. **Up to 60 minutes** after removal, the observer is still listed, but jobs are grayed out and marked offline until the next housekeeping process runs. You can delete existing jobs, but cannot view, add or edit jobs.
3. **After 60 minutes** the removed observer is no longer listed, **but jobs remain**, though they are grayed out and marked offline. You can delete existing jobs, but cannot view, add or edit jobs.
4. If **at any time** you reinstall or reactivate the observer, it reappears in the UI, and existing (previously active) jobs are no longer grayed out. You can delete, view or edit existing jobs, or add new jobs.

Configuring ALM Observer jobs

Using the Agile Lifecycle Manager Observer, you can define jobs that dynamically load data associated with intent from the Agile Lifecycle Manager for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the Agile Lifecycle Manager Kafka server host and topics to hand, such as the Agile Lifecycle Manager server, the Kafka port, and the topics used for lifecycle events.

Important: To access Agile Lifecycle Manager remotely, you must ensure that the Agile Lifecycle Manager installation has been configured with the **KAFKA_ADVERTISED_HOST_NAME** so as to allow remote connections. For more

information, see the Configuration reference topic in the Agile Lifecycle Manager Knowledge center at the following location: https://www.ibm.com/support/knowledgecenter/SS8HQ3_1.2.0/GettingStarted/r_alm_quickreference.html

The Agile Lifecycle Manager Observer is installed as part of the core installation procedure.

About this task

The Agile Lifecycle Manager Observer jobs listen to the Kafka 'state change' topics of Agile Lifecycle Manager, as well as the Agile Lifecycle Manager Resource Manager. Information is extracted from Agile Lifecycle Manager about Assemblies and Resources and a topology is created.

After installation, you define and start the following two jobs.

Listen for lifecycle events ('alm' job)
The **alm** job is a long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the observer is stopped.

Listen for Resource Manager events ('rm' job)
The **rm** job is a long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the observer is stopped.

Table 11. ALM Observer parameters for **alm** *jobs*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
IBM Agile Lifecycle Manager instance name	Use this to identify the Agile Lifecycle Manager installation and any associated Resource Managers.	Required
Topic	Use this to identify the Agile Lifecycle Manager Kafka topic.	Required
Group ID	Use this to identify the Kafka group ID to be used.	Required
Connection	Use this to specify the Kafka Host and Port to be used.	Required
Observer job description	Enter additional information to describe the job.	Optional

Table 12. ALM Observer parameters for ALM **rm** *(Resource Manager) jobs*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
IBM Agile Lifecycle Manager instance name	Use this to identify the Agile Lifecycle Manager installation and any associated Resource Managers.	Required

Table 12. ALM Observer parameters for ALM **rm** *(Resource Manager) jobs (continued)*

Parameter	Action	Details
Topic	Use this to identify the Agile Lifecycle Manager **Resource Manager** Kafka topic.	Required
Group ID	Use this to identify the Kafka group ID to be used.	Required
Connection	Use this to specify the Kafka Host and Port to be used.	Required
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the IBM Agile Lifecycle Manager icon, or select an existing ALM job to be edited.
2. Choose either **alm** or **rm** from the job type drop-down.

To configure an alm job

3. Enter or edit the following parameters:
   - Unique ID
   - IBM Agile Lifecycle Manager instance name
   - Topic (the Kafka topic for the Agile Lifecycle Manager lifecycle events)
   - Group ID
   - Connection

To configure an rm job

4. Enter or edit the following parameters:
   - Unique ID
   - IBM Agile Lifecycle Manager instance name
   - Topic (the Kafka topic for the Agile Lifecycle Manager Resource Manager lifecycle events)
   - Group ID
   - Connection

   Important: The value of the **`IBM Agile Lifecycle Manager instance name`** parameter needs to be the same for both jobs to allow for the topology to be combined.
5. Optional: Enter an **Observer job description** to explain the purpose of the job in more detail.
6. Click **Run job** to save your job and begin retrieving information.

Configuring AWS Observer jobs

Using the AWS Observer, you can define jobs that read services data from the Amazon Web Services (AWS) through AWS SDK and generate a topology. It is installed as part of the core installation procedure.

Before you begin

Ensure you have the AWS details to hand, such as AWS Region, Access Key ID and Access Secret Key.

About this task

The AWS Observer supports multiple Amazon web services such as EC2 for its 'elastic compute' services.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

Table 13. AWS Observer parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
accessKey	Specify the AWS access key.	Required
secretKey	Specify the AWS secret key.	Required. Must be encrypted.
region	Specify the AWS region to discover.	Required.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

The Load job requires the **secretKey** in the configuration file in encrypted form. To encrypt them, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the secret key. The encryption utility will return an encrypted **secretKey**.

Procedure

To find your Access Key and Secret Access Key:

1. Log in to your AWS Management Console.
2. Click on your user name at the top right of the page.
3. Click on the **Security Credentials** link from the drop-down menu.
4. Find the **Access Credentials** section, and copy the latest Access Key ID.
5. Click on the **Show link** in the same row, and copy the Secret Access Key.

To find the region

6. Check the region at the following location:
   https://docs.aws.amazon.com/general/latest/gr/rande.html

Note: The Full Topology Upload job only supports one region per full load. If you wish to discover more than one region, you will need to run multiple full loads.

To configure the AWS job

7. From the Observer Configuration UI, click **Configure** under the AWS icon, or select an existing job to be edited.
8. Enter or edit the following parameters:
   - Unique ID
   - Access key
   - Secret access key
   - Region
9. Optional: Enter an **Observer job description** to explain the purpose of the job in more detail.
10. Click **Run job** to save your job and begin retrieving information.

Configuring BigFix Inventory Observer jobs

You configure BigFix Inventory Observer jobs to read data from a Bigfix Inventory instance through its REST APIs, and generate a topology.

Before you begin

The Bigfix Inventory Observer is installed as part of the core installation procedure.

Before configuring a Bigfix Inventory job, ensure you have the Bigfix Inventory details to hand such as the Bigfix Inventory URL, API token and SSL trustStore.

Important: The Bigfix Inventory Observer supports Bigfix Inventory Version 9.5.0.

About this task

You define and start the following job.

Bigfix Inventory Observer job (full topology load)

A transient (one-off) job that loads a baseline of all requested topology data.

This job loads a baseline of topology data from an Bigfix Inventory environment.

Run this job whenever you need Bigfix Inventory topology data refreshed.

Table 14. Bigfix Inventory Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
BigFix Inventory API token	Enter the BigFix token for authentication.	Required. Must be encrypted.
BigFix Inventory instance URL	Specify the API URL of the BigFix Inventory endpoint (including port).	Required. Usually in the following format: `https://<hostname or IP address>:<port>`

Table 14. Bigfix Inventory Observer job parameters (continued)

Parameter	Action	Details
Bigfix Inventory resources	Specify the resources to be discovered.	Optional. Lists supported values such as software, hardware or *. If left blank, all available resources are discovered.
Bigfix Inventory certificate	Specify the name of the certificate to be loaded into the trust store.	Optional. If used, must be in the `/opt/ibm/netcool/asm/security` directory.
HTTPS trustStore file name	Specify the trustStore file name.	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
trustStore file password	Specify the trustStore password to decrypt the HTTPS trustStore file.	Required. Must be encrypted.
Bigfix Inventory connection timeout (ms)	Enter the time at which the connection times out.	Optional. Must be a value greater than 0 (zero), and the default is 5000 (5 seconds).
Data Center	Specify the data center(s) in which the Bigfix Inventory instance runs.	Required. If more than one, list them (comma-separated).
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement: The job requires passwords in encrypted form. To encrypt the Bigfix Inventory token and SSL trustStore file password, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

To configure and run Bigfix Inventory Observer jobs

1. From the Observer Configuration UI, click **Configure** under the Bigfix Inventory icon, or select an existing Bigfix Inventory job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - BigFix Inventory API token (must be encrypted)
   - BigFix Inventory instance URL
   - Bigfix Inventory resources (optional)
   - Bigfix Inventory certificate (optional)
   - HTTPS trustStore file name
   - trustStore file password (must be encrypted)
   - Bigfix Inventory connection timeout (ms) (optional)
   - data_center
   - Observer job description (optional)

3. Click **Run job** to save your job and begin retrieving information.

To acquire Bigfix Inventory SSL certificate and build SSL truststore

4. Use the following command to use OpenSSL to connect to Bigfix Inventory, and extract a SSL Certificate from Bigfix Inventory to a <*certificate_file_name*>.crt file.

   ```
   echo -n | openssl s_client -connect {Bigfix Inventory IpAddress}:{port} | sed
   -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
   ```

5. Use the following Java keytool command to import the Bigfix Inventory certificate file into a keystore and encrypt the keystore with a given password.

   ```
   keytool -import -v -trustcacerts -alias {Bigfix Inventory Hostname}
   -file {certificate_file_name}.crt -keystore {keystore file name}
   -storepass {your password to encrypt keystore}
   ```

6. Copy the keystore file ({keystore file name}) to the $ASM_HOME/security directory to complete the SSL setup.

What to do next

Run this job whenever you need Bigfix Inventory topology data refreshed.

Configuring Ciena Blue Planet Observer jobs

Using the Ciena Blue Planet Observer, you can define jobs that will gather and read all topology data from the Blue Planet MCP instance by REST API and generate a topology..

Before you begin

The Ciena Blue Planet Observer is installed as part of the core installation procedure.

Ensure you have the Ciena Blue Planet details to hand, such as API username, API password, MCP URL, MCP certificate, truststore file and truststore password.

About this task

The Ciena Blue Planet Observer has one job, which is the restapi job. When a restapi job is run, it loads baseline topology data through Blue Planet MCP APIs: Network Elements (constructs), EquipmentHolder, Equipment, TPE (Terminating Point Encapsulation), and FRE (Forwarding Relationship Encapsulation).

Tip: Defining observer jobs using the UI is the same for both on-premise and IBM Cloud Private.

After installation, you define and start the following job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

Table 15. Ciena Blue Planet Observer parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
username	MCP API username	Required
password	MCP API password	Required
tenant	The tenant to use	Required

Table 15. Ciena Blue Planet Observer parameters (continued)

Parameter	Action	Details
server_url	The URL of the MCP server instance	Required
mcp_certificate	Ciena BluePlanet MCP certificate	Optional. If used, must be in the `/opt/ibm/netcool/asm/security` directory.
ssl_trustorefile	Exact HTTPS trust store file name	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
ssl_truststore_password	The password to decrypt HTTPS trust store file	Required. Must be encrypted.
connect_read_timeout_ms	Sets the connection and read timeout in milliseconds	Optional. Must be a value greater than 0 (zero), and the default is 5000 (5 seconds).
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the Ciena Blue Planet icon, or select an existing Ciena Blue Planet job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - username
   - password
   - tenantserver_url
   - mcp_certificate
   - ssl_trustorefile
   - ssl_truststore_password (must be encrypted)
   - connect_read_timeout_ms
3. Optional: Enter an **Observer job description** to explain the purpose of the job in more detail.
4. Click **Run job** to save your job and begin retrieving information.

Configuring Cisco ACI Observer jobs

You use the Cisco ACI Observer when you have a Cisco ACI environment with Cisco Application Policy Infrastructure Controller (APIC) in your environment. The Observer interfaces with Cisco APIC and makes active REST calls to Cisco APIC in the Cisco ACI environment. You configure observer jobs that dynamically load Cisco ACI data for analysis by Netcool Agile Service Manager from the Observer Configuration UI.

Before you begin

Ensure you have the Cisco ACI service details to hand, such as the Cisco APIC username, Cisco APIC password, Cisco APIC SSL TrustStore and Cisco APIC URL.

The Cisco Application Centric Infrastructure (ACI) Observer is installed as part of the core installation procedure.

About this task

A Cisco ACI Observer job extracts Cisco ACI resources from Cisco APIC via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You define and start the following jobs.

restapi

A transient (one-off) job that loads all requested topology data using Cisco APIC REST APIs to build a tenant logical construct topology or a fabric topology, and then exits.

A 'restapi' job loads initial topology data, and can resynchronize topology data from Cisco ACI into the Agile Service Manager topology.

You assign 'restapi' as the job type for `/jobs/restapi` observer endpoint.

websocket

A long-running job that listens for notifications from Cisco APIC to build the topology and runs until it is explicitly stopped, or until the observer is stopped.

A 'websocket' job monitors changes from Cisco APIC object notification and updates the Agile Service Manager topology.

You always run a 'websocket' job **after** running a 'restapi' job type.

You assign 'websocket' as the job type for `/jobs/websocket` observer endpoint.

Table 16. Cisco ACI Observer **restapi** *and* **websocket** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
Cisco APIC password	Enter the password for Cisco APIC authentication.	Required. Must be in encrypted text.
Cisco APIC endpoint	Specify the API URL of the Cisco APIC endpoint.	Required. Usually in the following format: `https://`*`[hostname or IP address]`*`/api`
Cisco APIC certificate	Specify a certificate by name to load into the trustStore.	Optional. Must be located in the `/opt/ibm/netcool/asm/security` directory.
HTTPS trustStore file name	Specify the trustStore file name.	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
HTTPS trustStore file password	Specify the trustStore password to decrypt the HTTPS trustStore file.	Required
Cisco APIC username	Specify the username to connect as, or listen to.	Required
Tenant name	Use this to identify the tenant.	Required. Set to 'admin' if there is no specific tenant. Set to '' to load Fabric Topology resources.

Table 16. Cisco ACI Observer **restapi** *and* **websocket** *job parameters (continued)*

Parameter	Action	Details
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

Both jobs require passwords in encrypted form. To encrypt the password and file name, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

To acquire a Cisco APIC SSL certificate and build the SSL truststore, see the instructions from step six onwards of the following procedure.

Procedure

To configure Cisco ACI Observer jobs

1. From the Observer Configuration UI, click **Configure** under the Cisco ACI icon, or select an existing Cisco ACI job to be edited.
2. Choose either **restapi** or **websocket** from the job type drop-down.
3. Enter or edit the following parameters for both job types:
   - Unique ID
   - Cisco APIC password (must be encrypted)
   - Cisco APIC endpoint
   - Cisco APIC certificate (optional)
   - HTTPS trustStore file name
   - HTTPS trustStore file password (must be encrypted)
   - Cisco APIC username
   - Tenant name
4. Optional: Enter an **Observer job description** to explain the purpose of the job in more detail.
5. Click **Run job** to save your job and begin retrieving information.

To acquire a Cisco APIC SSL certificate and build the SSL truststore

6. Required: For **ICP** Agile Service Manager deployments, use the relevant instructions in the following topic: “Defining observer security” on page 49
7. Required: For **on-prem** Agile Service Manager deployments, use the relevant instructions in the following topic: Defining Cisco ACI Observer jobs (on-prem)

Configuring Contrail Observer jobs

Using the Contrail Observer, you can retrieve topology data from Juniper Network Contrail Release 4.1 via REST APIs exposed by the Contrail API server. This observer is developed against Juniper Network Contrail that integrates with OpenStack orchestration platform (Ubuntu 16.04 + Contrail Cloud - Ocata).

Before you begin

Ensure you have the Contrail API Server and OpenStack credentials details to hand. For rabbitmq jobs, you must also specify the location of the RabbitMQ queue and its authentication details.

The Contrail observer is installed as part of the core installation procedure.

About this task

Contrail Observer jobs retrieve topology data from Juniper Network Contrail Release 4.1 via REST APIs exposed by the Contrail API server. The observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You define and start the following jobs.

rabbitmq
A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

This job loads all supported resources during startup, and listens to RabbitMQ messages from 'vnc_config.object-update' fanout exchange.

There is no need to run the restapi job before running the rabbitmq job, because the rabbitmq job performs a restapi job during initialization.

restapi
A transient (one-off) job that loads all requested topology data.

This job loads all supported resources.

Run this job whenever you need the Contrail topology data refreshed.

Table 17. Contrail Observer **rabbitmq** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
RabbitMQ hostname or IP address	Enter a hostname or IP address for the RabbitMQ server.	Required
RabbitMQ port	Specify the port for connection.	Required
RabbitMQ username	Specify the username for authentication with RabbitMQ.	Required
RabbitMQ password	Enter the password for authentication with RabbitMQ.	Required. Must be encrypted.
RabbitMQ virtual host	Enter the RabbitMQ virtual hostname	Optional. Default is /.

Table 17. Contrail Observer **rabbitmq** *job parameters (continued)*

Parameter	Action	Details
Contrail API URL	Specify the URL for the Contrail API server.	Required
OpenStack Authentication URL	Enter the authentication URL for the identity service.	Required
OpenStack username	Specify the OpenStack user name to connect as (or to).	Required
OpenStack password	Specify the OpenStack password with which to authenticate.	Required
Authentication type	Specify the authentication type used.	Optional. The default is **Keystone**, the other option is **None**.
OpenStack project domain name	Specify the OpenStack project domain name.	Optional
OpenStack user domain name	Specify the OpenStack project user domain name.	Optional
OpenStack project name	Enter the OpenStack project name for version 3 authentication	Optional
OpenStack tenant name	Enter the OpenStack tenant name for version 2 authentication	Optional
OpenStack identity API version	Select an option from the dropdown list.	Optional
Observer job description	Enter additional information to describe the job.	Optional

Table 18. Contrail Observer **restapi** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
Contrail API URL	Specify the URL for the Contrail API server.	Required
OpenStack Authentication URL	Enter the authentication URL for the identity service.	Required
OpenStack username	Specify the OpenStack user name to connect as (or to).	Required
OpenStack password	Specify the OpenStack password with which to authenticate.	Required
Authentication type	Specify the authentication type used.	Optional. The default is **Keystone**, the other option is **None**.
OpenStack project domain name	Specify the OpenStack project domain name.	Optional
OpenStack user domain name	Specify the OpenStack project user domain name.	Optional

Table 18. Contrail Observer **restapi** *job parameters (continued)*

Parameter	Action	Details
OpenStack project name	Enter the OpenStack project name for version 3 authentication	Optional
OpenStack tenant name	Enter the OpenStack tenant name for version 2 authentication	Optional
OpenStack identity API version	Select an option from the dropdown list.	Optional
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

Both jobs require the Contrail token in encrypted form. To encrypt the token, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the Contrail icon, or select an existing Contrail job to be edited.
2. Choose either **restapi** or **rabbitmq** from the job type drop-down.

To configure a restapi job

3. Enter or edit the following **required** parameters:
   - Unique ID
   - Contrail API URL
   - Openstack Authentication URL
   - Openstack username
   - Openstack password
4. Enter or edit the following **optional** parameters:
   - Authentication type
   - OpenStack project domain name
   - OpenStack user domain name
   - OpenStack project name
   - OpenStack tenant name
   - OpenStack identity API version
   - Observer job description

To configure a rabbitmq job

5. Enter or edit the following parameters:
   - Unique ID
   - RabbitMQ hostname or IP address
   - RabbitMQ port
   - RabbitMQ username
   - RabbitMQ password (must be encrypted)

- RabbitMQ virtual host (**optional**)
- Contrail API URL
- Openstack Authentication URL
- Openstack username
- Openstack password

6. Enter or edit the following **optional** parameters:
   - Authentication type
   - OpenStack project domain name
   - OpenStack user domain name
   - OpenStack project name
   - OpenStack tenant name
   - OpenStack identity API version
   - Observer job description
7. Click **Run job** to save your job and begin retrieving information.

Configuring DNS Observer jobs

Using the DNS Observer, you can query internal DNS server performance, and use the returned information on response times and service addresses to create topologies within the topology service. The DNS Observer supports forward and reverse job types, with 'recurse' or 'no recurse' options.

Before you begin

Ensure you have the DNS access details to hand, such as DNS server, address types and port numbers.

The DNS Observer is installed as part of the core installation procedure.

About this task

The DNS Observer provides DNS query services and topological insight into how a specified DNS server is performing forward (name-to-IP address) or reverse (IP address-to-name) lookups. Query results include a list of addresses, information on how long it takes the DNS server to resolve a lookup, and, optionally (with the maximum number of recursive calls set at 200) how the DNS server is recursively resolving a given name or IP address.

Tip: The relationship types can be customized with line color, width and pattern functions. See the “Creating custom relationship type styles” on page 202 topic for more information.

You define and start the following jobs.

Reverse lookup job
: A transient (one-off) job that loads all requested DNS reverse (IP address-to-name) lookup topology data.

Forward lookup job
: A transient (one-off) job that loads all requested DNS forward (name-to-IP address) lookup topology data.

Table 19. DNS Observer **reverse** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
ip_address	Enter the host or internet name to lookup.	Required
Address Types	Specify the address types to be observed.	Required. Select either IPv4 or IPv6.
Server	Specify the DNS server.	Required
Port	Specify the UDP DNS port.	Optional
Recursive query	Toggle **True** or **False**.	Optional. If set to True, the maximum number of calls is set at 200.
Observer job description	Enter additional information to describe the job.	Optional

Table 20. DNS Observer **forward** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
Domain name	Specify a domain.	Required
Address Types	Specify the address types to be observed.	Required. Select either IPv4 or IPv6.
Server	Specify the DNS server.	Required
Port	Specify the UDP DNS port.	Optional
Recursive query	Toggle **True** or **False**.	Optional. If set to True, the maximum number of calls is set at 200.
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the DNS icon, or select an existing DNS job to be edited.
2. Choose either **reverse** or **forward** from the job type drop-down.

Configure a 'reverse' job

3. Enter or edit the following required parameters:
   - Unique ID
   - ip_address
   - Address Types
   - ServerPort (optional)
   - Recursive query (optional)
   - Observer job description (optional)

Configure a 'forward' job

4. Enter or edit the following parameters:
   - Unique ID
   - Domain name

- Address Types
- ServerPort (optional)
- Recursive query (optional)
- Observer job description (optional)

5. Click **Run job** to save your job and begin retrieving information.

Configuring Docker Observer jobs

Using the Docker Observer, you can discover Docker network resources, including Docker Swarm clusters, and then visualize (or model) this data as a topology view in the Agile Service Manager UI. You configure observer jobs from the Observer Configuration UI.

Before you begin

Ensure you have the details for your Docker job to hand, specifically your Docker system's Unix socket, and / or host and port number.

The Docker Observer is installed as part of the core installation procedure.

About this task

Using the Observer Configuration UI you configure observer jobs that query the Docker REST API to retrieve data and display it as a topology in the Topology Viewer. The Docker Observer can model external Docker systems, and it can also provide a System health view of the Docker system on which Agile Service Manager runs.

The job parameters determine whether to connect to a local Docker on the same (UNIX) host as the observer using the **unix_socket** parameter, or to a remote Docker using the **host** and **port** parameters.

Table 21. Docker Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Host	Use this to identify the TCP host socket (HTTP or HTTPS) on which to access the remote Docker system.	Required for **remote** Docker access only
Username	Specify the username of the remote Docker environment with HTTPS.	Required for remote Docker with HTTPS access only.
Password	Specify the password of the remote Docker environment with HTTPS.	Required for remote Docker with HTTPS access only. Must be encrypted.
Docker SSL Certificate	Specify the certificate file name.	Optional
Docker SSL TrustStore File	Specify the trustStore file name.	Required for remote Docker with HTTPs access only.
SSL TrustStore File Password	Specify the trustStore password.	Required for remote Docker with HTTPs access only. Must be encrypted.

Table 21. Docker Observer job parameters (continued)

Parameter	Action	Details
Port	Use this to identify the TCP port (HTTP or HTTPS) on which to access the remote Docker system.	Required for **remote** Docker access only
Unix Socket	Use this to access local docker environments using the complete path.	Required for **local** Docker access only. Host and port parameters **must** be empty.
View	Use this to select which resources are modeled in the topology view.	Optional. The Default displays running resources only. Options are: **Container** All running containers **Image** Images used by running containers **Task** Running tasks only
Containers to exclude	List container you want to exclude.	Optional
Job description	Use this to describe the job in greater detail	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the Docker icon, or select an existing Docker job to be edited.
2. Configure one of the following job types:
   - To discover **remote** Docker network resources **through TCP port exposure**, enter or edit the following parameters:
     - Unique ID
     - Host
     - Port
     - View (optional)
     - Containers to exclude (optional)
     - Job description (optional)
   - To discover **remote** Docker network resources **through HTTPS**, enter or edit the following parameters:
     - Unique ID
     - Host
     - Port
     - Username
     - Password
     - Docker SSL Certificate (optional)
     - Docker SSL TrustStore File
     - SSL TrustStore File Password
     - View (optional)
     - Containers to exclude (optional)

- Job description (optional)

- To discover **local** Docker networks (if the Unix socket is accessible via the Docker container), enter or edit the following parameters:
  - Unique ID
  - Unix socket
  - View (optional)
  - Containers to exclude (optional)
  - Job description (optional)

  Restriction: For local Docker networks, the **host** and **port** parameter fields must be empty.

3. Click **Run job** to save your job and begin retrieving information.

Configuring Dynatrace Observer jobs

Using the Dynatrace Observer, you can query a specified Dynatrace environment for information about its applications, services, process groups, and infrastructure entities.

Before you begin

Ensure you have generated a Dynatrace token to access your Dynatrace environment. You also need topology access scope to access the Dynatrace resources.

Ensure you have the Dynatrace access details to hand, such as Dynatrace API URL and API token.

The Dynatrace Observer is installed as part of the core installation procedure.

About this task

You define and start the following job.

Dynatrace job
A transient (one-off) job that loads all requested Dynatrace resource data.

Table 22. Dynatrace Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
Base URL	Specify the API URL of the Dynatrace endpoint (including version).	Required
API Token	Enter the Dynatrace token for authentication.	Required. Must be encrypted.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement: The job requires the API token in encrypted form. To encrypt the Dynatrace token token, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the Dynatrace icon, or select an existing Dynatrace job to be edited.
2. Enter or edit the following job parameters:
   - Unique ID
   - Base URL
   - API Token
   - Observer job description (optional)
3. Click **Run job** to save your job and begin retrieving information.

Configuring Event Observer jobs

You use the Event Observer to get events from Netcool/OMNIbus, via the XML Gateway, into the Agile Service Manager topology service. Netcool/OMNIbus events can also be generated from Agile Service Manager status via the Netcool/OMNIbus Message Bus probe.

Before you begin

The Event Observer is installed as part of the core installation procedure.

ICP Note: Most prerequisites are deployed and configured during installation.

- This **includes** the Netcool/OMNIbus XML Gateway and the Netcool/OMNIbus Message Bus probe, which you **do not** have to configure before defining Event Observer jobs.
- In addition, when the ASM chart is deployed a listen job is automatically created, which tells the Event Observer to listen to the object servers deployed by the NOI chart. This job is listed when you open the Observer Configuration UI (**DASH** > **Administration** > **Observer jobs** > **Existing jobs**).

On-prem Note:

Most prerequisites are deployed during the Agile Service Manager core installation. This **excludes** the Netcool/OMNIbus XML Gateway and the Netcool/OMNIbus Message Bus probe, which you must download, install and configure separately, but **includes** the Event Observer docker container, which has been installed and should be running, as well as the required scripts to manage jobs. Before you define an Event Observer job, you **must** configure the Netcool/OMNIbus XML Gateway and the Netcool/OMNIbus Message Bus probe, as described in the following topics:

- "Deploying the XML Gateway for Event Observer" on page 23
- "Deploying the Netcool/OMNIbus probe for Message Bus" on page 30

The Event Observer requires:

- Netcool/OMNIbus Probe for Message Bus **Version 8** or later
- Netcool/OMNIbus XML Gateway for Message Bus **Version 9** or later

About this task

The Event Observer receives batches of Netcool/OMNIbus events from the XML gateway. If it finds any matching resources in the Agile Service Manager topology service, it sets their status according to this event data, which you can then visualize in the Agile Service Manager UI.

The Event Observer runs a single long-running job for each tenant. This job listens for Netcool/OMNIbus events, which it receives via the Netcool/OMNIbus XML Gateway, and then sets the status of resources in the topology service.

Table 23. Event Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Netcool Sources	Specify the objects servers from which to obtain data.	Required (comma-separated list). To accept input from all object servers, set to '*' (asterisk).
Field names	Enter a (comma-separated) list of Netcool/OMNIbus ObjectServer `alerts.status` field names to identify top-level resources.	Optional. If empty, defaults to `"Node, NodeAlias"`. If **not** empty, all alerts.status field names must be listed. **Tip:** You can define extra event properties to be added to the status displayed in the Topology Viewer using the `extra_status_fields` property. Add a list of names, such as 'sourceId' or 'name'. You can then define topology viewer status tools that reference these.
Thread limit	Enter the number of received events to be processed in parallel.	Optional. The default is 100.
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the Event icon, or select an existing Event Observer job to be edited.
2. Enter or edit (at least) the following parameters:
   - Unique ID
   - Netcool sources
3. Click **Run job** to save your job and begin retrieving information.

Results

The Event Observer job monitors the Netcool/OMNIbus XML Gateway for updates and runs until it is stopped, or until the observer is stopped.

Related tasks:

"Configuring custom tools" on page 192
As an administrator or advanced user, you can create custom topology tools, which users can then access from within a topology's context menu. This functionality allows you to access properties of a selected item (such as a resource or relationship) and execute some custom functionality within the context of that item.

Configuring File Observer jobs

Using the File Observer functionality, you can write bespoke data to a file in a specific format, upload this data to the topology service, and then visualize this data as a topology view in the Agile Service Manager UI.

Before you begin

The File Observer is installed as part of the core installation procedure.

About this task

The File Observer reads topology data from files located in the `$ASM_HOME/data/file-observer/` directory, and uploads it. You must create these files manually.

Topology data in a file is comprised of vertices (nodes) and edges. A vertex represents an object (resource), while an edge represents the relationship between two objects.

Each line of the file you create should be in one of the formats below, loading a single resource vertex (including optional relationships in the _references field) or a single edge, deleting a single vertex, or pausing execution.

Lines starting with `V:` (vertex), `E:` (edge), `D:` (delete) or `W:` (wait) are treated as instruction lines to be processed. Other lines, for example lines that are empty or commented out, are ignored.

Line format

V: Load a resource vertex, with a JSON representation as documented for the body of the topology service API method: `POST /resources`

If specifying the **`_status`** element, acceptable state values are open, closed, or clear, and acceptable severity values are clear, indeterminate, warning, minor, major, or critical.

E: Load an edge, with a JSON representation as documented for the **`_references`** section of the body of the topology service API method `POST /resources`

D: Delete a resource vertex, identified by it's uniqueId

W: Pause for the given duration (for testing purposes only).

Takes an integer period followed by a string specifying the units.

Tip: An example file is available in the `$ASM_HOME/data/file-observer` directory.

Table 24. File Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
File Name	Specify the name of the file to be loaded.	Required. Must be relative to the `$ASM_HOME/data/file-observer/` directory (rather than absolute).
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the File icon, or select an existing File job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - File Name
   - Observer job description (optional)
3. Click **Run job** to save your job and begin retrieving information.

Results

The File Observer job loads all requested topology data from the file specified. This job runs only once.

What to do next

Run this job when the content in your file has been updated.

Configuring IBM Cloud Observer jobs

Use the IBM Cloud Observer when you have IBM Cloud installed in your environment to run jobs that read data from an IBM cloud instance. These jobs retrieve Cloud Foundry Apps information and services, and then dynamically load the retrieved data for analysis by Netcool Agile Service Manager.

Before you begin

Important: The IBM Cloud Observer supports Cloud Foundry API version 2.92.

Ensure you have the IBM Cloud access details to hand to specify and access the cloud instance, such as the instance ID, credentials, and region.

The IBM Cloud Observer is installed as part of the core installation procedure.

About this task

You define and start the following job.

Full Topology Upload job
: A transient (one-off) job that loads all requested topology data.

  The IBM Cloud Observer imports ITSM Resource Topology Service data to Agile Service Manager.

Table 25. IBM Cloud Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Instance	Enter the name of the IBM cloud instance.	Required
User email	The email address used to access the instance.	Required
Encrypted password	Enter the password used to access the instance.	Required. Must be in encrypted text.
IBM Cloud Region	Choose the cloud instance region from the drop-down list: • US_S • UK • EU • AP	Required. Each region has its own URI, and only a single region is discovered in a full load job. To discover different regions, a full load job needs to be triggered for each region.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

Both jobs require passwords in encrypted form. To encrypt the password and file name, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the IBM Cloud icon, or select an existing IBM Cloud job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - Instance
   - User email
   - Encrypted password
   - IBM Cloud Region
   - Observer job description (optional)
3. Click **Run job** to save your job and begin retrieving information.

Results

The IBM Cloud Observer job loads all requested topology data. Run this job whenever you need the IBM Cloud topology data refreshed.

Configuring Kubernetes Observer jobs

Using this observer, you can configure jobs that discover the structure of your Kubernetes clusters, including pods, worker nodes and containers.

Before you begin

For Kubernetes load jobs, ensure you have the Kubernetes service details to hand, such as the Kubernetes host IP and SSL Certificate details. For Weave Scope listen jobs, first install Weave Scope, and then configure a job using the Weave Scope IP and port parameters.

The Kubernetes Observer is installed as part of the core installation procedure.

Update Note:

From Agile Service Manager 1.1.5 onwards, the location of the Weave Scope listen job changes. As a result, existing Weave Scope jobs that were running during an upgrade to Version 1.1.5 will have their paths renamed when the observer starts. However, Weave Scope jobs that were not running (stopped) will not be recognized and so will not have their path renamed. As a consequence, the UI will be unable to restart them.
Workaround: Ensure that your Weave Scope jobs are running before you update your system.

About this task

The observer reads topology data from Kubernetes through its REST APIs, or Weave Scope.

You can run the following jobs:

Load A transient (one-off) job that loads all requested topology data from a Kubernetes or IBM Cloud Private environment.

weave_scope
A standalone job that listens to the Weave Scope agent and continues to stream topology and state data to Agile Service Manager.

The Weave Scope listen job provides visibility of your Kubernetes services, pods, containers, deployments, stateful sets, Cron Jobs and processes for a specified namespace.

A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped

You must install Weave Scope and then use the Weave Scope Master IP and Node port parameters. For more information on Weave Scope, see the following location: https://www.weave.works/docs/scope/latest/introducing/

For IBM Cloud Private v 2.1.0.3

1. Install Weave Scope as in the following example:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/scope.yaml?k8s-service-type=NodePort&k8s-version=$(kubectl version | base64 | tr -d '\n')"
   ```

   The installation creates a port (NodePort) that the Kubernetes Observer can use.
2. Identify the port using the following command:

```
kubectl -n weave describe service weave-scope-app
```

3. Launch the Weave Scope User Interface:

   ```
   http://<master ip>:<NodePort>
   ```

4. If the UI is empty or you are experiencing connection issues, you check the pod and agent using the following options:

   kubectl get -n weave pods
   Gets all the pods from the weave namespace.

   The weave scope app pod should be running.

   kubectl get -n weave daemonsets
   Gets all daemonsets from the weave namespace.

   There should be a weave-scope-agent running per host in the Kubernetes cluster.

   kubectl describe -n weave daemonsets weave-scope-agent
   This command describes the weave scope agent daemonset.

   If the value for weave-scope-agent in the daemonsets is 0 (zero), a security error appears at the end in the events section.

   In the case of a security error, create the following configuration files:

   PodSecurityPolicy
   Example:

   ```
   apiVersion: extensions/v1beta1
   kind: PodSecurityPolicy
   metadata:
     name: weave-scope
   spec:
     privileged: true
     hostPID: true
     hostNetwork: true
     allowedCapabilities:
     - 'NET_ADMIN'
     fsGroup:
       rule: RunAsAny
     runAsUser:
       rule: RunAsAny
     seLinux:
       rule: RunAsAny
     supplementalGroups:
       rule: RunAsAny
     volumes:
     - '*'
   ```

   ClusterRole
   Example:

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRole
   metadata:
     name: weave-scope
   rules:
   - apiGroups:
     - extensions
     resourceNames:
     - weave-scope
     resources:
     - podsecuritypolicies
     verbs:
     - use
   ```

ClusterRoleBinding

Example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: weave-scope-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: weave-scope
subjects:
- kind: ServiceAccount
  name: weave-scope
  namespace: weave
```

kubectl apply -f <filename>

This command applies each of the configuration file.

On starting, the weave-scope-agent should now be ready and available.

For IBM Cloud Private 3.1.1

1. Create Namespace 'weave' with 'ibm-privileged-psp'.

   ```
   kubectl create namespace weave
   kubectl -n weave create rolebinding weave-clusterrole-rolebinding --
   clusterrole=ibm-privileged-clusterrole --group=system:serviceaccounts:
   weave
   ```

2. Install Weave Scope using the following command:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/scope.yaml?k8s-service-
   type=NodePort&k8s-version=$(kubectl version | base64 | tr -d '\n')"
   ```

   This will result in a port being opened that the Observer can use.

3. You can discover the NodePort using the following command:

   ```
   kubectl -n weave describe service weave-scope-app
   ```

4. Launch the Weave Scope UI using the following URL:

   ```
   http://<master ip>:<NodePort>
   ```

Table 26. Kubernetes Observer **load** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Encrypted Kubernetes token	The service account token for kubernetes.	Required. Must be encrypted.
Kubernetes Master IP address	Enter the Kubernetes Master IP address.	Required
Kubernetes API port	Enter the Kubernetes API port number.	Required
Exact HTTPS certificate file name	Enter the exact name of the SSL/HTTPS certificate.	Required
data_center	Specify the name of the data center in which the Kubernetes instance is running.	Required
Namespace	Specify the Kubernetes namespace.	Optional. If left empty, all namespaces are observed.

Table 26. Kubernetes Observer **load** *job parameters (continued)*

Parameter	Action	Details
API query timeout (ms)	Specify the Kubernetes REST API query timeout.	Optional. The default is 5000 ms (that is, 5 seconds)
Terminated pods	Choose whether terminated pods should be hidden (**true** or **false**).	Optional. The default is false.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement: The Load job requires the token to be encrypted. You encrypt the Kubernetes token using the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password, which you enter in the **Encrypted Kubernetes token** field when configuring the Load job.

SSL certificate requirement: The Load job requires an SSL Certificate, and for it to be in a specific location:

1. Get the kubernetes master IP and its API port using:

   ```
   kubectl cluster-info
   ```

2. Run the following OpenSSL command:

   ```
   echo -n | openssl s_client -connect {master ip}:{api} | sed -ne
   '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./certificate_file_name.crt
   ```

   The certificate is saved as *`certificate_file_name.crt`*

3. Copy the certificate file to the `$ASM_HOME/security` directory.
4. When configuring the Load job, enter the certificate file name in the **Exact HTTPS certificate file name** field.

Table 27. Kubernetes Observer **weave_scope** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Host	Enter the Weave Scope host name (or IP address) of the web socket to be observed.	Required
Port	Enter the Weave Scope port number of the web socket to be observed.	Required
Cluster Name	Enter the name of the cluster or data center to be observed.	Required
Namespaces	Enter a list of namespaces to be observed.	Optional. If left empty, all namespaces will be observed.
Resource types	Select the Weave Scope resource types to observe.	Optional.
Resources to exclude	List resources to be excluded by ID, label, rank or namespace.	Optional. Containers named 'pod' are excluded by default.

Table 27. Kubernetes Observer **weave_scope** *job parameters (continued)*

Parameter	Action	Details
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the Kubernetes icon, or select an existing Kubernetes job to be edited.
2. Choose either **load** or **weave_scope** from the job type drop-down.

Configure the load job

3. Enter or edit the following **required** parameters:
   - Unique ID
   - Encrypted Kubernetes token
   - Kubernetes Master IP address
   - Kubernetes API port
   - Exact HTTPS certificate file name
   - data_center
4. Enter or edit the following **optional** parameters:
   - Namespaces
   - API query timeout (ms)
   - Terminated pods
   - Observer job description

Configure the weave_scope job

5. Enter the following **required** parameters:
   - Unique ID
   - Host
   - Port

     Tip: The NodePort can be obtained using the following command:

     ```
     kubectl -n weave describe service weave-scope-app
     ```
   - Cluster Name

   Note: The **host** and **port** parameter fields must be empty.
6. Enter the following **optional** parameters:
   - Namespaces

     Tip: Run the following command in the Kubernetes environment to get a list of namespaces:

     ```
     kubectl get namespaces
     ```
   - Resource types
   - Resources to exclude
   - Observer job description
7. Click **Run job** to save your job and begin retrieving information.

Configuring Network Manager Observer jobs

Using the IBM Tivoli Network Manager (ITNM) Observer, you can define jobs that dynamically load data discovered by Network Manager for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the IBM Tivoli Network Manager service details to hand, such as the domain, host and port number.

The ITNM Observer is installed as part of the core installation procedure.

Update Note:

When updating from Agile Service Manager Version 1.1.3 to a later version, you must run a migration script to avoid the creation of duplicate ITNM Observer records **before** running any ITNM Observer jobs:

```
cd $ASM_HOME/bin
/execute_crawler.sh -c itnm_provider_transfer
```

- Running this script **before** making any new observations with the ITNM Observer prevents the creation of duplicate records.
- Running this script **after** making new observations with the ITNM Observer removes duplicate records, but **may not** preserve some historical topology data previously gathered by the ITNM Observer.

The script, which may take some time to complete on large topologies, creates a management artifact in the topology. You can monitor its progress by querying the artifact via Swagger.

About this task

The ITNM Observer jobs extract Network Manager resources using an Object Query Language JDBC driver. The Observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

You configure the following two jobs.

Load A transient (one-off) job that queries Network Manager for topology data, and performs a complete upload for a single ITNM domain.

Listen A long-running job that monitors the Network Manager message bus for changes and update the topology service accordingly. When the job is started, the observer creates an OQL connection that listen for changes in the ITNM network. Any resources added, changed or deleted are passed on by the OQL connection and the Agile Service Manager topology service is updated.

The listen job runs until it is explicitly stopped, or until the observer is stopped.

Table 28. ITNM Observer **load** *and* **listen** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required

Table 28. ITNM Observer **load** *and* **listen** *job parameters (continued)*

Parameter	Action	Details
ITNM instance name	Specify the ITNM instance name.	Required. Specify the same instance name for all jobs to enable connectivity across domains.
ITNM domain	Specify the ITNM domain.	Required
Hostname or Server IP	Specify the ITNM host name or server IP on which the domain is running.	Required
ITNM domain port	Specify the ITNM port for the specified domain.	Required. For more information, see Tip (ITNM port).
OQL connection timeout (ms)	Specify the OQL Connection timeout value.	Optional. The default is 3000 (30 seconds).
Exclude resources without connections	Select whether to display disconnected resources.	Optional. The choices are **true** and **false**. The default is **true**.
Edge Type Map	Map ITNM layers to topology relationship types.	Optional. If left blank, Agile Service Manager will auto-map. For more information, see the edge type mapping tip.
Observer job description	Enter additional information to describe the job.	Optional

Tip (ITNM port): The value of **port** will vary if multiple domains exist. To identify which port is associated with a specific domain in your Network Manager host, open the `$NCHOME/etc/precision/ServiceData.cfg` file and locate the line that specifies which ncp_config service binds to the domain, for example:

```
SERVICE: ncp_config DOMAIN: NCOMS ADDRESS: 172.17.0.4 PORT: 7968 SERVERNAME:
core.ibm.com DYNAMIC: NO
```

The **port** identified in this example is `7968` (while the **domain** is `NCOMS`, and the **host** (ITNM Server IP) is `172.17.0.4`).

Tip (edge type mapping): To identify topology relationship types, see the following file: `$NCHOME/precision/disco/stitchers/DNCIM/PopulateDNCIMTopologies.stch` Alternatively, run the following OQL statement against the model service to list the available topology types:

```
select ENTITYNAME from ncimCache.entityData where METACLASS='Topology'
```

Procedure

1. From the Observer Configuration UI, click **Configure** under the ITNM icon, or select an existing ITNM job to be edited.
2. Choose either **load** or **listen** from the job type drop-down.
3. Configure the following **required** parameters for both **load** and **listen** jobs:
   - Unique ID
   - ITNM instance name
   - ITNM domain
   - Hostname or Server IP

- ITNM domain port

4. Configure the following **optional** parameters for both **load** and **listen** jobs:
   - OQL connection timeout (ms)
   - Exclude resources without connections
   - Edge Type Map
   - Observer job description
5. Click **Run job** to save your job and begin retrieving information.

Configuring New Relic Observer jobs

Use New Relic Observer when you have a New Relic account with a New Relic Infrastructure subscription. Using New Relic Observer, you can configure jobs that dynamically load New Relic Infrastructure resource data via New Relic for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the New Relic account and New Relic Infrastructure subscription details to hand, such as the account name, account ID, and New Relic Insights API query key.

The New Relic Observer is installed as part of the core installation procedure.

Restriction: New Relic applies a 1000 results limit on all New Relic Query Language (NRQL) queries. To accommodate this limit when retrieving data from the SystemSample, StorageSample, ProcessSample and NetworkSample event tables, the New Relic Observer uses the following NRQL query time clause:

```
"SINCE 4 hours ago LIMIT 1000"
```

About this task

The Observer uses the New Relic Infrastructure subscription and makes active New Relic Query Language (NRQL) calls over REST to New Relic Insights to download New Relic Infrastructure resource data.

The New Relic Observer loads the following New Relic Infrastructure resources and their relationships to the Agile Service Manager core topology service:

- Host
- Storage
- OS
- Network Interfaces
- Processes

The New Relic Observer job extracts New Relic Infrastructure resources from New Relic using New Relic Query Language (NRQL) over REST. The observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

You configure the following job.

Load A transient (one-off) job that loads all requested topology data.

Table 29. New Relic job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
New Relic Name	Specify the New Relic account name or tenant name.	Required
New Relic account ID	Specify the New Relic account ID.	Required. For more information, see account ID tip
New Relic Insights Query API key	Specify the New Relic Insights Query API key.	Required. Must be encrypted. For more information, see query API key tip
filterCriteria	Extend the result set returned to Agile Service Manager.	Optional. The default value is 'SINCE 4 hours ago LIMIT 1000'. For more information, see the documentation for New Relic Query Language.

Tip (New Relic account ID): To obtain the account ID, first log into the New Relic login page:
https://login.newrelic.com/login and then obtain the account ID from this URL:
https://rpm.newrelic.com/accounts/%3CaccountId%3E

Tip (New Relic Insights Query API key): A new Relic user with a new Relic Infrastructure subscription is required to generate a new Relic Insights query API Key as outlined here: https://docs.newrelic.com/docs/insights/insights-api/get-data/query-insights-event-data-api

Encryption requirement

The load job requires the `insightsQueryAPIKey` in encrypted form. To encrypt the insightsQueryAPIKey, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the New Relic icon, or select an existing New Relic job to be edited.
2. Configure (at least) the following parameters:
   - Unique ID
   - New Relic account name or tenant name
   - New Relic account ID
   - New Relic Insights Query API Key (must be encrypted)
3. Click **Run job** to save your job and begin retrieving information.

Results

This job loads all requested topology data. Run this job whenever you need New Relic topology data refreshed.

Configuring OpenStack Observer jobs

Using the OpenStack Observer, you can configure jobs that dynamically load OpenStack data for analysis by Agile Service Manager.

Before you begin

Ensure you have the OpenStack service details to hand, such as the parameters for its APIs or RabbitMQ message bus. If you are configuring a query job, have OpenStack location and authorisation details to hand. If you are configuring a rabbitmq job, you must also identify and provide access to the RabbitMQ message bus.

OpenStack installation requirements:

If you have installed OpenStack using DevStack, you must add the code specified here to the end of the `local.conf` file, and reinstall it. If you have installed OpenStack using another installation method, you must add the code specified here to the `nova.conf` file, and then restart the Nova (compute) service.

If you have already installed OpenStack using DevStack
Add the following code to the end of the `local.conf` file, and then reinstall OpenStack.

If you are planning to install OpenStack using DevStack
Add the following code to the end of the `local.conf` file before installation.

```
[[post-config|$NOVA_CONF]]
[DEFAULT]
notification_topics = notifications,com.ibm.asm.obs.nova.notify
notification_driver=messagingv2
notify_on_state_change=vm_and_task_state
notify_on_any_change=True
```

For standard (or any other) OpenStack installations
Add the following code under the `[DEFAULT]` section of the `nova.conf` file, and then restart the Nova (compute) service.

```
notification_topics = notifications,com.ibm.asm.obs.nova.notify
notification_driver=messagingv2
notify_on_state_change=vm_and_task_state
notify_on_any_change=True
```

The OpenStack Observer is installed as part of the core installation procedure.

About this task

The OpenStack Observer jobs extract OpenStack resources via REST or RabbitMQ. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You configure and run the following two jobs.

restapi

A transient (one-off) job that loads all requested topology data from the OpenStack instance by REST API.

The job loads baseline topology data through the following OpenStack's APIs:

- Keystone (identity)
- Cinder (block storage)
- Glance (image)
- Heat (orchestration)
- Neutron (network)
- Nova (compute)

Restriction: An OpenStack environment that has a list of endpoints whereby the heat-cfn service comes first before the heat service, will encounter a JSON parsing error in the logs due to a known issue in the openstack4j library. When this happens, the full load for the heat service will be skipped entirely. The other service will run as normal.

rabbitmq

A long-running job that reads messages on OpenStack's RabbitMQ message bus for activity from the Cinder (block storage), Heat (orchestration), Neutron (network) and Nova (compute) components continually, until it is explicitly stopped, or until the Observer is stopped.

The rabbitmq job should only be run after an initial restapi job has been completed.

Restriction: Only one rabbitmq job should be listening to one queue (or sets of queues) at any one time. If you need to listen to multiple projects, then separate queues must be set up in OpenStack, with appropriate names, before separate listening jobs are submitted for each. For example, for Nova via the **rmq_nova_notify** attribute, for Neutron via the **rmq_neutron_notify** attribute.

Table 30. OpenStack Observer **restapi** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
OpenStack authentication type	Specify the OpenStack connection authentication technique to use.	Required. Choose either **V2_Tenant**, **V3_Unscoped**, **V3_Project**, **V3_Domain**, or **V3_ProjectDomain**.
OpenStack password	Specify the OpenStack password with which to authenticate.	Required. Must be encrypted.
OpenStack identity endpoint	Specify the authentication URL.	Required. Must include the port and version.
Data center name	Specify the name of the data center in which the OpenStack instance is running.	Required. If more than one OpenStack instance is run, and duplicate project or tenant names exist, you must disambiguate them here.
OpenStack username	Specify the OpenStack user name to connect as (or to).	Required

Table 30. OpenStack Observer **restapi** *job parameters (continued)*

Parameter	Action	Details
OpenStack tenant name	Specify the OpenStack tenant.	Required.
OpenStack project name	Specify the OpenStack project.	Optional
OpenStack domain name	Specify the OpenStack domain name.	Optional.
OpenStack region name	Specify the OpenStack region.	Optional
OpenStack perspective	Select the URL perspective the API accesses data from.	Optional. Choose from **Admin**, **Public**, and **Internal**.
Connection and read timeout (ms)	Choose the timeout setting for the connection and read actions.	Optional. The default is 5000 (5 seconds).
SSL Verification	Choose whether to use SSL verification.	Optional. Choose **true** or **false**. If false, HTTPS is used, but **without** a certificate.
Observer job description	Enter additional information to describe the job.	Optional

Table 31. OpenStack Observer **rabbitmq** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
RabbitMQ username	Specify the AMQP user name to connect to the broker.	Required
RabbitMQ password	Specify the password to use to connect to the broker.	Required. Must be encrypted.
RabbitMQ hosts	Enter a (comma-seperated) list of hosts in the RabbitMQ cluster.	Required. The first successful connection is used.
Data center name	Specify the name of the data center in which the OpenStack instance is running.	Required. If more than one OpenStack instance is run, and duplicate project or tenant names exist, you must disambiguate them here.
OpenStack username	Specify the OpenStack user name to connect as (or to).	Required
OpenStack tenant name	Specify the OpenStack tenant.	Required
OpenStack project name	Specify the OpenStack project.	Optional
RabbitMQ virtual host name	Specify the virtual host to connect to the broker.	Optional
Use SSL?	Choose whether to use an SSL connection.	Optional. Choose **true** or **false**. For RabbitMQ, you must choose **true**.
Nova v2 Oslo message queue	Specify the Nova v2 Oslo message queue.	Optional

Table 31. OpenStack Observer **rabbitmq** *job parameters (continued)*

Parameter	Action	Details
Neutron v2 Oslo message queue	Specify the Neutron v2 Oslo message queue.	Optional
Cinder v2 Oslo message queue	Specify the Cinder v2 Oslo message queue.	Optional
Heat v2 Oslo message queue	Specify the Heat v2 Oslo message queue.	Optional
Number of consumer instances	Specify the number of consumer instances to create for each API queue type.	Optional
Observer job description	Enter additional information to describe the job.	Optional

Important: You **must** specify the following properties consistently for both the restapi and rabbitmq jobs:

- Data center name
- OpenStack tenant name
- OpenStack project name
- OpenStack username

Encryption requirement:

The restapi and rabbitmq jobs require passwords in the configuration file to be encrypted. To encrypt the OpenStack or RabbitMQ passwords, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password, for example:

```
2IuExvqz5SGnGgROYGLAQg==
```

Procedure

1. From the Observer Configuration UI, click **Configure** under the OpenStack icon, or select an existing OpenStack job to be edited.
2. Choose either **restapi** or **rabbitmq** from the job type drop-down.

Configure the OpenStack Observer restapi job

3. Enter or edit the following **required** parameters:
   - Unique ID
   - OpenStack authentication type
   - OpenStack password (must be encrypted)
   - OpenStack identity endpoint
   - Data Center name
   - OpenStack username
   - OpenStack tenant name
4. Enter or edit the following **optional** parameters:
   - OpenStack project name
   - OpenStack domain name
   - OpenStack region name

- OpenStack perspective
- Connection and read timeout (ms)
- SSL Verification
- Observer job description

Configure the OpenStack Observer rabbitmq job

5. Enter or edit the following parameters:
   - Unique ID
   - RabbitMQ username
   - RabbitMQ password (must be encrypted)
   - RabbitMQ hosts
   - Data center name
   - OpenStack username
   - OpenStack tenant name
6. Enter or edit the following **optional** parameters:
   - OpenStack project name
   - RabbitMQ virtual host name
   - Use SSL?
   - Nova v2 Oslo message queue
   - Neutron v2 Oslo message queue
   - Cinder v2 Oslo message queue
   - Heat v2 Oslo message queue
   - Number of consumer instances
   - Observer job description
7. Click **Run job** to save your job and begin retrieving information.

Configuring REST Observer jobs

Use the REST Observer for obtaining topology data via REST endpoints. This observer is a counterpart to the File Observer.

Before you begin

The REST (or RESTful) Observer is installed as part of the core installation procedure.

About this task

The REST Observer passes topology data to Agile Service Manager using a RESTful set of interfaces, which provide REST APIs that enable the following functionality:

- Management of Listen and bulk-replace job types.
- The insert-update (HTTP POST) of resources.
- The insert-update (HTTP POST) of relationships.
- The insert-replacement (HTTP PUT) of resources.
- The deletion (HTTP DELETE) of resources.
- A REST API that supports the deletion (HTTP DELETE) of all relationships of a given type from a specified resource.
- A REST API that supports the deletion (HTTP DELETE) of a specific relationship.

Restriction: Resources created via REST can have a provider, but not an observer.

Benefits

Using the REST Observer rather than the File Observer or Topology Service APIs includes the following benefits:

- The ability to provide data to Agile Service Manager via HTTP REST Endpoints instead of files.
- The processing performed by all observers in their framework ensures that meta-data about observations from observers is managed correctly.
- A simple way of deleting all edges of a given type on a resource or a specific edge instance.

To use the REST Observer, a job request must be issued (HTTP POST) to the Observer instance job management APIs before sending data to the Resource and Relationship APIs.

Listen A long-running listen job capable of consuming topology data over a long period of time.

A listen job is designed to support scenarios where the input data stream is unpredictable, or there is little or no consistency or versioning of resources within the data stream.

Synchronize (bulk replace, or load)

A long-running job with the same resource replace semantics as the File Observer.

Bulk-replace jobs are designed to support scenarios where a known set of resources are subject to updates or versioning, and a prior observation about resources is to be replaced with a new one.

This job can provide a new set of resources and relationships and synchronize them to Agile Service Manager, thereby causing any previous data provided by the Observer to be deleted and replaced with the new data.

Note: In both cases, a provider job parameter is required to identify the origin of the data being provided to the Observer job.

Once a job request has been successfully submitted, you can start to provide data to the Resource and Relationship APIs on behalf of a given job instance.

The Resource and Relationship APIs may respond with an `HTTP 503 Service Unavailable` response with a `Retry-After: 10 seconds` in the header. This indicates that even though the request against those APIs is valid, the observer has not been able to ascertain that meta-data about the job is held in Agile Service Manager yet; this may be due to, for example, any prevailing conditions in the network that support the Agile Service Manager micro-services.

Tip: If such a response is received, try the request again later.

Table 32. REST Observer **listen** *and* **bulk replace** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required

Table 32. REST Observer **listen** *and* **bulk replace** *job parameters (continued)*

Parameter	Action	Details
Provider	Specify the name of the program or system to provide data.	Required
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the REST icon, or select an existing REST job to be edited.
2. Choose either **listen** or **bulk replace** from the job type drop-down.
3. Configure the following parameters for both **bulk replace** and **listen** jobs:
   - Unique ID
   - Provider
   - Observer job description (optional)
4. Click **Run job** to save your job and begin retrieving information.

Configuring ServiceNow Observer jobs

Using the ServiceNow Observer job, you can retrieve the configuration management database (CMDB) data from ServiceNow via REST API, using basic authentication credentials.

Before you begin

Ensure your user account has the `rest_api_explorer` and `web_service_admin` roles. These roles are required to access the resources from ServiceNow. Also, ensure you have the ServiceNow service details to hand, such as username, password, and URL.

The ServiceNow Observer is installed as part of the core installation procedure.

About this task

ServiceNow jobs retrieve configuration management database (CMDB) data from ServiceNow via REST API. The Observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

You define and start the following job.

ServiceNow job
A transient (one-off) job that loads all requested topology data.

Run this job whenever you want to refresh the ServiceNow topology data.

Table 33. ServiceNow Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
ServiceNow instance	Specify the ServiceNow instance.	Required

Table 33. ServiceNow Observer job parameters (continued)

Parameter	Action	Details
ServiceNow username	Specify the ServiceNow username.	Required
ServiceNow password	Specify the ServiceNow password.	Required. Must be encrypted.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

The ServiceNow job requires the password in the configuration file to be encrypted. To encrypt the password, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the ServiceNow icon, or select an existing ServiceNow job to be edited.
2. Configure the following parameters for the ServiceNow job:
   - Unique ID
   - ServiceNow instance
   - ServiceNow username
   - ServiceNow password (must be encrypted)
   - Observer job description
3. Click **Run job** to save your job and begin retrieving information.

What to do next

Run this job whenever you want to refresh the ServiceNow topology data.

Configuring TADDM Observer jobs

TADDM Observer jobs retrieve network topology data (including discovered applications, their components, configurations and dependencies) from the TADDM database server (running either a IBM DB2 or an Oracle database), and use this data to create topologies within the Agile Service Manager topology service.

Before you begin

To connect to a TADDM Oracle database, you must place the Oracle JDBC Driver into the `$ASM_HOME/lib` folder, and then restart the observer for it to take effect. You can download the driver from the Oracle website, or copy it them from the Oracle server (**not** the Oracle client) from the following location: `../app/oracle/product/Oracle_version/dbhome/jdbc/lib/ojdbc6.jar`

Ensure you have the TADDM Rest API login access details in hand, such as the TADDM API URL, username and password.

The TADDM Observer is installed as part of the core installation procedure.

About this task

TADDM Observer jobs retrieve topology data using the TADDM REST API. The observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You define and start the following jobs.

Load job
A transient (one-off) job that loads all requested topology data.

Table 34. TADDM Observer **load** *job parameters*

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
TADDM API URL	Specify the TADDM endpoint to connect to.	Required.
TADDM username	Specify the TADDM user name.	Required
TADDM password	Specify the password for the TADDM user.	Required. Must be encrypted.
TADDM objects to observe	Select one or more options from the drop-down list.	Optional. If none are selected, all supported model objects are retrieved.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

The Load job requires passwords in the configuration file to be encrypted. To encrypt the password, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

1. From the Observer Configuration UI, click **Configure** under the TADDM icon, or select an existing TADDM job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - TADDM API URL
   - TADDM username
   - TADDM password (must be encrypted)
   - TADDM objects to observe (optional)
   - Observer job description (optional)
3. Click **Run job** to save your job and begin retrieving information.

Configuring VMware NSX Observer jobs

You configure VMware NSX Observer jobs to dynamically load data from the VMware NSX REST interface.

Before you begin

You can use the VMware NSX Observer when you have a VMware NSX appliance in your environment.

Important: The VMware NSX Observer supports VMware NSX versions 6.2 and 6.3.

Ensure you have the VMware NSX REST API details to hand, such as the VMware NSX URL, username, password, and SSL trustStore.

The VMware NSX Observer is installed as part of the core installation procedure.

About this task

The VMware NSX Observer job extracts VMware NSX resource information via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You define and start the following job.

VMware NSX Observer job (full topology load)

A transient (one-off) job that loads a baseline of all requested topology data.

This job loads a baseline of topology data from an environment which contains a VMware NSX appliance.

Run this job whenever you need VMware NSX topology data refreshed.

The VMware NSX Observer loads the following resources and their relationship into the Netcool Agile Service Manager core topology service:

- NSX Appliance
- vCenter Appliance
- NSX Controller
- Edge Router - Logical (Distributed) Router, Edge Service Gateway
- Virtual Machines
- Host
- VNIC

Table 35. VMware NSX Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
Network Virtualization and Security Platform password	Enter the password to authenticate with.	Required. Must be encrypted.
Network Virtualization and Security Platform API URL	Specify the API URL of the VMware NSX endpoint.	Required. Usually in the following format: `https://[hostname or IP address]/api`

Table 35. VMware NSX Observer job parameters (continued)

Parameter	Action	Details
SSL trustStore file	Specify the trustStore file name.	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
SSL trustStore file password	Specify the trustStore password to decrypt the HTTPS trustStore file.	Required.
Connection and read timeout (ms)	Enter the time at which the connection and read actions time out.	Optional. Must be a value greater than 0 (zero), and the default is 5000 (5 seconds).
Data center name	Specify the data center in which the VMware NSX instance runs.	Required.
Network Virtualization and Security Platform username	Specify the username to connect as, or listen to.	Required
Network Virtualization and Security Platform tenant name	Specify the tenant.	Optional.
Network Virtualization and Security Platform certificate	Specify a certificate by name to load into the trustStore.	Optional. Must be located in the `/opt/ibm/netcool/asm/security` directory.
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement: The job requires passwords in encrypted form. To encrypt the VMware NSX password (nsx_password) and SSL trustStore file password (password_ssl_truststore_file), run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

To configure and run VMware NSX Observer jobs

1. From the Observer Configuration UI, click **Configure** under the VMware NSX icon, or select an existing VMware NSX job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - Network Virtualization and Security Platform password
   - Network Virtualization and Security Platform API URL
   - SSL trustStore file
   - SSL trustStore file password (must be encrypted)
   - Connection and read timeout (ms) (optional)
   - Data center name
   - Network Virtualization and Security Platform username
   - Network Virtualization and Security Platform tenant name (optional)

- Network Virtualization and Security Platform certificate (optional)
- Observer job description (optional)

3. Click **Run job** to save your job and begin retrieving information.

To acquire VMware NSX SSL certificate and build SSL truststore

4. Required: For **ICP** Agile Service Manager deployments, use the relevant instructions in the following topic: "Defining observer security" on page 49
5. Required: For **on-prem** Agile Service Manager deployments, use the relevant instructions in the following topic: Defining VMware NSX Observer jobs (on-prem)

What to do next

Run this job whenever you need VMware NSX topology data refreshed.

Configuring VMware vCenter Observer jobs

You configure VMware vCenter Observer jobs to dynamically load data from the VMware vCenter REST interface.

Before you begin

Important: The VMware vCenter Observer supports integration with VMware vCenter versions 6.5 and 6.7.

Ensure you have the VMware vCenter service details to hand, such as username, password, SSL TrustStore and URL.

The VMware vCenter Observer is installed as part of the core installation procedure.

About this task

The VMware vCenter Observer job extracts VMware vCenter resource information via REST. The Observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

You define and start the following job.

VMware vCenter Observer job (full topology load)
A transient (one-off) job that loads a baseline of all requested topology data.

Run this job whenever you need VMware vCenter topology data refreshed.

The VMware vCenter Observer loads the following resources and their relationship into the Agile Service Manager core topology service:

- ESXi / ESX Hosts
- Virtual Machines
- VNICs
- Storage

Table 36. VMware vCenter Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required

Table 36. VMware vCenter Observer job parameters (continued)

Parameter	Action	Details
vCenter password	Enter the password to authenticate with.	Required. Must be encrypted.
vCenter API URL	Specify the API URL of the VMware vCenter endpoint (including port and version).	Required. Usually in the following format: `https://[hostname or IP address]/rest`
HTTPS trustStore file name	Specify the trustStore file name.	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
trustStore file password	Specify the trustStore password to decrypt the HTTPS trustStore file.	Required.
Data center name	Specify the data center in which the VMware vCenter instance runs.	Required. If more than one, list them (comma-separated).
vCenter username	Specify the username to connect as, or listen to.	Required
vCenter certificate	Specify a certificate by name to load into the trustStore.	Optional. Must be located in the `/opt/ibm/netcool/asm/security` directory.
Connection and read timeout	Enter the time at which the connection and read actions time out.	Optional. Must be a value greater than 0 (zero), and the default is 5000 (5 seconds).
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement: The job requires passwords in encrypted form. To encrypt the VMware vCenter password (vcenter_password) and SSL trustStore file password (password_ssl_truststore_file), run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

Procedure

To configure and run VMware vCenter Observer jobs

1. From the Observer Configuration UI, click **Configure** under the VMware vCenter icon, or select an existing VMware vCenter job to be edited.
2. Enter or edit the following parameters:
   - Unique ID
   - vCenter password
   - vCenter API URL
   - HTTPS trustStore file name
   - trustStore file password (must be encrypted)
   - Data center name
   - vCenter username

- vCenter certificate (optional)
- Connection and read timeout (optional)
- Observer job description (optional)

3. Click **Run job** to save your job and begin retrieving information.

To acquire VMware vCenter SSL certificate and build SSL truststore

4. Required: For **ICP** Agile Service Manager deployments, use the relevant instructions in the following topic: "Defining observer security" on page 49
5. Required: For **on-prem** Agile Service Manager deployments, use the relevant instructions in the following topic: Defining VMware vCenter Observer jobs (on-prem)

What to do next

Run this job whenever you need VMware vCenter topology data refreshed.

Configuring Zabbix Observer jobs

Using the Zabbix Observer functionality, you can load monitored servers and their associated network resources, and then visualize this data as a topology view in the Agile Service Manager UI. It is installed as part of the core installation procedure.

Before you begin

The Zabbix Observer supports Zabbix Version 4.0.3.

Ensure you have the Zabbix server details to hand, such as the username, password and SSL TrustStore.

About this task

A Zabbix Observer job extracts server information and its associated network resources from Zabbix via REST RPC. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

You define and start the following job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

Table 37. Zabbix Observer parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
The datacenter of zabbix server	Specify the data center in which the Zabbix instance runs.	Required. If more than one, list them (comma-separated).
The hostname of zabbix server	Enter the Zabbix virtual hostname.	Required
The username of zabbix server	Specify the Zabbix username.	Required
The password of zabbix server	Specify the Zabbix user password.	Required. Must be encrypted.

Table 37. Zabbix Observer parameters (continued)

Parameter	Action	Details
Zabbix ssl certificate	Specify a certificate file name.	Optional. If provided then a certificate file with the same name must exist in the `$ASM/security` directory.
HTTPS trustStore file name	Specify the trustStore file name.	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
trustStore file password	Specify the trustStore password to decrypt the HTTPS trustStore file.	Required. Must be encrypted.
Zabbix connection timeout (milliseconds) (optional)	Timeout, in ms, when querying the Zabbix REST API.	Optional. Default is 5000 (5s).
Observer job description	Enter additional information to describe the job.	Optional

Procedure

1. From the Observer Configuration UI, click **Configure** under the Zabbix icon, or select an existing Zabbix job to be edited.
2. Configure the following parameters for the Zabbix job:
   - Unique ID
   - The datacenter of zabbix server
   - The hostname of zabbix server
   - The username of zabbix server
   - The password of zabbix server (must be encrypted)
   - Zabbix ssl certificate (optional)
   - HTTPS trustStore file name
   - trustStore file password (must be encrypted)
   - Zabbix connection timeout (milliseconds) (optional)
   - Observer job description (optional)
3. Click **Run job** to save your job and begin retrieving information.

To acquire Zabbix SSL certificate and build SSL truststore

4. Required: For **ICP** Agile Service Manager deployments, use the relevant instructions in the following topic: “Defining observer security” on page 49
5. Required: For **on-prem** Agile Service Manager deployments, use the relevant instructions in the following topic: Defining Zabbix Observer jobs (on-prem)

Observer reference

Before observers can load data, you must first define and then run observer jobs. This section describes how to **manually** configure, schedule and run observers.

Remember: It is recommended that you use the Observer Configuration UI to create and run observer jobs, instead of editing job configuration files manually, as described here. However, to schedule jobs and configure trust stores and certificates, you can use the information in this section.

All prerequisites are deployed during the Agile Service Manager core installation. This includes the docker containers for the observers, which should be installed and running, as well as the required scripts to manage jobs.

You can verify that the observer docker containers are running using the following command:

```
/opt/ibm/netcool/asm/bin/asm_status.sh
```

The system will return text showing all running containers in a state of `Up`.

Observer jobs are configured and run from the Observer Configuration UI, and can be long-running or transient. For example, the Network Manager Observer topology 'load' job is a one-off, transient job, while the Network Manager and Event Observer 'listen' jobs are long-running, which run until explicitly stopped, or until the Observer is stopped.

In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **`-h`** or **`--help`** to display help information, and with **`-v`** or **`--verbose`** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining ALM Observer jobs

Using the Agile Lifecycle Manager Observer, you can define jobs that dynamically load data associated with intent from the Agile Lifecycle Manager for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the Agile Lifecycle Manager Kafka server host and topics to hand, such as the Agile Lifecycle Manager server, the Kafka port, and the topics used for lifecycle events.

Important: To access Agile Lifecycle Manager remotely, you must ensure that the Agile Lifecycle Manager installation has been configured with the **`KAFKA_ADVERTISED_HOST_NAME`** so as to allow remote connections. For more information, see the Configuration reference topic in the Agile Lifecycle Manager Knowledge center at the following location: https://www.ibm.com/support/knowledgecenter/SS8HQ3_1.2.0/GettingStarted/r_alm_quickreference.html

About this task

The Agile Lifecycle Manager Observer jobs listen to the Kafka 'state change' topics of Agile Lifecycle Manager, as well as the Agile Lifecycle Manager Resource Manager. Information is extracted from Agile Lifecycle Manager about Assemblies and Resources and a topology is created.

`alm_observer_common.sh`
: The configuration file you use to customize the listening job for the Agile Lifecycle Manager lifecycle events topic.

  The parameters defined here are then used by the `alm_observer_listen_start.sh` script to trigger the Agile Lifecycle Manager Observer job.

alm_observer_common_rm.sh
: The configuration file you use to customize the listening job for the Agile Lifecycle Manager Resource Manager lifecycle events topic.

  The parameters defined here are then used by the `alm_observer_listen_start_rm.sh` script to trigger the Agile Lifecycle Manager Observer job.

After installation, you define and start the following two jobs. You must edit the parameters in the configuration file before running these jobs.

Listener for Agile Lifecycle Manager lifecycle events
: A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the observer is stopped.

  This job is started by the `alm_observer_listen_start.sh` script.

Listener for Agile Lifecycle Manager Resource Manager lifecycle events
: A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the observer is stopped.

  This job is started by the `alm_observer_listen_start_rm.sh` script.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/alm-observer/swagger`

Procedure

1. Edit (at least) the following parameters in the `alm_observer_common.sh` configuration file:

   `connection`
   : The host and port of the Agile Lifecycle Manager Kafka server.

2. Edit (at least) the following parameters in the `alm_observer_common_rm.sh` configuration file:

   `topic` The Kafka topic for the Agile Lifecycle Manager Resource Manager lifecycle events.

   `connection`
   : The host and port of the Agile Lifecycle Manager Kafka server.

   Note: The value of the **`almInstall`** parameter needs to be the same for both jobs to allow for the topology to be combined.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

3. To start the Agile Lifecycle Manager Observer **Listener for Agile Lifecycle Manager lifecycle events** job, use the following command:

   ```
   $ASM_HOME/bin/alm_observer_load_start_rm.sh
   ```

   The Listener for Agile Lifecycle Manager lifecycle events job monitors its source for updates and runs until it is stopped, or until the Observer is stopped.

4. To start the Agile Lifecycle Manager Observer **Listener for Agile Lifecycle Manager Resource Manager lifecycle events** job, use the following command:

   ```
   $ASM_HOME/bin/itnm_observer_listen_start_rm.sh
   ```

   The Listener job monitors its source for updates and runs until it is stopped, or until the Observer is stopped.

What to do next

You can also use the following scripts:

alm_observer_listen_stop.sh
: This script stops the Listener job for Agile Lifecycle Manager lifecycle events.

alm_observer_listen_stop_rm.sh
: This script stops the Listener job for Agile Lifecycle Manager Resource Manager lifecycle events.

alm_observer_job_list.sh
: This script lists the current job status.

alm_observer_log_level.sh
: This script sets the log level.

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining AWS Observer jobs

Using the AWS Observer, you can define jobs that read services data from the Amazon Web Services (AWS) through AWS SDK and generate a topology. It is installed as part of the core installation procedure.

Before you begin

Ensure you have the AWS details to hand, such as AWS Region, Access Key ID and Access Secret Key.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/aws-observer/swagger`

About this task

The AWS Observer supports multiple Amazon web services such as EC2 for its 'elastic compute' services.

aws_observer_common.sh
The configuration file you use to customize AWS Observer settings.

The parameters defined here are then used by the `aws_observer_load_start.sh` to trigger the AWS Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `aws_observer_load_start.sh` script.

Required: In order for the AWS Observer to access the Amazon Web Services (AWS) account automatically, the **accessKey**, **secretKey** and **region** parameters are required.

- The access key and the secret access key are not the standard user name and password, but are special tokens that allow the services to communicate with the AWS account by making secure REST or Query protocol requests to the AWS service API.
- The region is the geographical location, for example **US East (Ohio)**, **Asia Pacific (Hong Kong)**, or **EU (London)**.

  Note: The Full Topology Upload job only supports one region per full load. If you wish to discover more than one region, you will need to run multiple full loads.

Procedure

To find your Access Key and Secret Access Key:

1. Log in to your AWS Management Console.
2. Click on your user name at the top right of the page.
3. Click on the **Security Credentials** link from the drop-down menu.
4. Find the **Access Credentials** section, and copy the latest Access Key ID.
5. Click on the **Show link** in the same row, and copy the Secret Access Key.

To find the region

6. Check the region at the following location:
   https://docs.aws.amazon.com/general/latest/gr/rande.html

To edit the parameters in the configuration file

7. Open the `aws_observer_common.sh` configuration file and edit (at least) the following Load parameters:

   accessKey
   AWS access key

   secretKey
   AWS secret key

region AWS region to discover.

Encryption requirement:

The Load job requires the **secretKey** in the configuration file in encrypted form. To encrypt them, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the secret key. The encryption utility will return an encrypted **secretKey**.

To start the Load job

8. To start the AWS Observer Full Topology Upload job, use the following command:

```
$ASM_HOME/bin/aws_observer_load_start.sh
```

Results

This job loads all requested topology data, and runs only once. Run this job whenever you need AWS topology data refreshed.

What to do next

You can also use the following scripts:

aws_observer_load_stop.sh
Stops the Load job

aws_observer_job_list.sh
Lists the status of current jobs

aws_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining BigFix Inventory Observer jobs

The Bigfix Inventory Observer is installed as part of the core installation procedure. Using the Bigfix Inventory Observer, you can define jobs that dynamically load Bigfix inventory data for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the Bigfix Inventory service details to hand, such as API token, SSL TrustStore and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/bigfixinventory-observer/swagger`

About this task

The Bigfix Inventory Observer jobs extract Bigfix Inventory resources via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

bigfixinventory_observer_common.sh
The configuration file you use to customize Bigfix Inventory Observer settings.

The parameters defined here are then used by the `bigfixinventory_observer_load_start.sh` to trigger the Bigfix Inventory Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `bigfixinventory_observer_load_start.sh` script.

Procedure

To edit the parameters in the configuration file

1. Open the `bigfixinventory_observer_common.sh` configuration file and edit (at least) the following Load parameters:

   instance_url
   BigFix Inventory instance URL of the BigFix Inventory endpoint (including port)

   Usually in the following format: `https://<hostname or IP address>:<port>`

   api_token
   Bigfix Inventory API token

   resources
   Bigfix Inventory resources to discover.

   Values are 'software', 'hardware', or '*' (for both)

   truststore_file
   Bigfix Inventory SSL trust store file for HTTPS authentication

   truststore_password
   Password to decrypt and encrypt Bigfix Inventory SSL trust store file

   data_center
   The data center(s) in which the Bigfix Inventory instance runs.

   Note: The default data center specified in `bigfixinventory_observer_common.sh` is sufficient for a default deployment. If you require more than one custom data centers, enter them as a comma-separated list.

   Encryption requirement:

The Load job requires the **api_token** and **truststore_password** in the configuration file in encrypted form. To encrypt them, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the token and password. The encryption utility will return an encrypted **api_token** and **truststore_password**.

To start the Load job

2. To start the Bigfix Inventory Observer Full Topology Upload job, use the following command:

```
$ASM_HOME/bin/bigfixinventory_observer_load_start.sh
```

Results

This job loads all requested topology data, and runs only once. Run this job whenever you need Bigfix Inventory topology data refreshed.

What to do next

You can also use the following scripts:

bigfixinventory_observer_load_stop.sh
Stops the Load job

bigfixinventory_observer_job_list.sh
Lists the status of current jobs

bigfixinventory_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Ciena Blue Planet Observer jobs

Using the Ciena Blue Planet Observer, you can define jobs that will gather and read all topology data from the Blue Planet MCP instance by REST API and generate a topology.

Before you begin

The Ciena Blue Planet Observer is installed as part of the core installation procedure.

Ensure you have the Ciena Blue Planet details to hand, such as API username, API password, MCP URL, MCP certificate, truststore file and truststore password.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/cienablueplanet-observer/swagger`

About this task

The Ciena Blue Planet Observer has one job, which is the restapi job. When a restapi job is run, it loads baseline topology data through Blue Planet MCP APIs: Network Elements (constructs), EquipmentHolder, Equipment, TPE (Terminating Point Encapsulation), and FRE (Forwarding Relationship Encapsulation).

Tip: Defining observer jobs using the UI is the same for both on-premise and IBM Cloud Private.

`cienablueplanet_observer_common.sh`
The configuration file you use to customize Ciena Blue Planet Observer settings.

The parameters defined here are then used by the `cienablueplanet_observer_load_start.sh` to trigger the Ciena Blue Planet Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `cienablueplanet_observer_load_start.sh` script.

Procedure

To edit the parameters in the configuration file

1. Open the `cienablueplanet_observer_common.sh` configuration file and edit the following parameters:

Table 38. Ciena Blue Planet Observer parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job.	Required
username	MCP API username	Required
password	MCP API password	Required
tenant	The tenant to use	Required
server_url	The URL of the MCP server instance	Required
mcp_certificate	Ciena BluePlanet MCP certificate	Optional. If used, must be in the `/opt/ibm/netcool/asm/security` directory.
ssl_trustorefile	Exact HTTPS trust store file name	Required. The supported format is JKS and the file is relative to `$ASM_HOME/security`
ssl_truststore_password	The password to decrypt HTTPS trust store file	Required. Must be encrypted.

Table 38. Ciena Blue Planet Observer parameters (continued)

Parameter	Action	Details
connect_read_timeout_ms	Sets the connection and read timeout in milliseconds	Optional. Must be a value greater than 0 (zero), and the default is 5000 (5 seconds).
Observer job description	Enter additional information to describe the job.	Optional

Encryption requirement:

The Load job requires the **password** and the **ssl_truststore_password** in the configuration file in encrypted form. To encrypt them, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and confirm the password. The encryption utility returns passwords in encrypted format.

To start the Load job

2. To start the Ciena Blue Planet Observer Full Topology Upload job, use the following command:

```
$ASM_HOME/bin/cienablueplanet_observer_load_start.sh
```

Results

This job loads all requested topology data, and runs only once. Run this job whenever you need Ciena Blue Planet topology data refreshed.

What to do next

You can also use the following scripts:

cienablueplanet_observer_load_stop.sh
Stops the Load job

cienablueplanet_observer_job_list.sh
Lists the status of current jobs

cienablueplanet_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Cisco ACI Observer jobs

The Cisco Application Centric Infrastructure (ACI) Observer is installed as part of the core installation procedure. You use the Cisco ACI Observer when you have a Cisco ACI environment with Cisco Application Policy Infrastructure Controller (APIC) in your environment. The Observer interfaces with Cisco APIC and makes active REST calls to Cisco APIC in the Cisco ACI environment. Using the Cisco ACI Observer, you can define jobs that dynamically load Cisco ACI data for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the Cisco ACI service details to hand, such as the Cisco APIC username, Cisco APIC password, Cisco APIC SSL TrustStore and Cisco APIC URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/ciscoaci-observer/swagger`

About this task

A Cisco ACI Observer job extracts Cisco ACI resources from Cisco APIC via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`ciscoaci_observer_common.sh`
The configuration file you use to customize Cisco ACI Observer settings.

The parameters defined here are then used by the `ciscoaci_observer_query_start.sh` script to trigger the Cisco ACI Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following jobs. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `ciscoaci_observer_query_start.sh` script.

Listener
A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

This job is started by the ciscoaci_observer_listen_start.sh script.

The Cisco ACI Observer loads the following Cisco ACI objects and their relationships into the Netcool Agile Service Manager core topology service:

```
Tenant Logical construct
(1) fvTenant
(2) fvAp
    A policy owner in the virtual fabric
(3) fvAEPg
    A set of requirements for the application-level EPG instance
(4) fvAEpP
    Abstract representation of an endpoint profile
(5) fvEpP
    An endpoint profile
(6) fvBD
    A bridge domain is a unique layer 2 forwarding domain that contains one or more subnets
```

```
(7) fvCtx
    The private layer 3 network context that belongs to a specific tenant or is shared
(8) vzBrCP
    A contract is a logical container for the subjects which relate to the filters that
govern the rules for communication between endpoint groups (EPGs)
(9) vzOOBBrCP
    An out-of-band binary contract profile can only be provided by an out-of-band endpoint
group and can only be consumed by the external prefix set
(10) vzSubj
    A subject is a sub-application running behind an endpoint group (for example, an
Exchange server). A subject is parented by the contract, which can encapsulate multiple subjects
(11) vzFilter
    A filter policy is a group of resolvable filter entries
(12) fvSubnet
    A subnet defines the IP address range that can be used within the bridge domain
(13) fvRsCons
    The Consumer contract profile information and on Cisco ACI gui the option to create
this object is via Consumed Contract. Used to build relationship between fvAEPg and vzBrCP
(14) fvRsBd
    A source relation to the bridge domain associated to this endpoint group. Used to build
relationship between fvBD and fvAEPg
(15) fvRsCtx
    A source relation to a private layer 3 network context that either belongs to a specific
tenant or is shared. Used to build relationship between fvBD and fvCtx
(16) vzRsSubjFiltAtt
    The filter for the subject of a service contract. Used to build relationship between
vzSubj and vzFilter

Fabric Topology
(1) fabricInst
    A container object for fabric policies
(2) fabricNode
    The root node for the APIC
(3) polUni
    Represents policy definition or resolution universe
(4) firmwareRunning
    Information about leaf or spine switch firmware running on a node
(5) firmwareCtrlrRunning
    Information about each controller firmware that is running
(6) eqptLCSlot
    The slot for the module card
(7) eqptLC
    A line card (IO card) contains IO ports
(8) eqptPsuSlot
    The power supply slot
(9) eqptPsu
    The power supply unit
(10) eqptFtSlot
    A fan tray slot
(11) eqptFan
    The fan in a fan tray
(12) topSystem
    Used to retrieve fabric node Operational State
(13) cnwPhysIf
    The physical interface assigned to the node cluster
(14) l1PhysIf
    The object that represents the Layer 1 physical Ethernet interface information object
(15) mgmtMgmtIf
    The management interface
(16) lldpAdjEp
    The LLDP neighbors, which contains the information regarding the neighbors
(17) eqptRsIoPhysConf
    A source relation to an L1 Ethernet interface. Used to build relationship between
l1PhysIf and eqptLC
(18) mgmtRsOoBStNode
    An object which contains management ip address of fabric spine switches and fabric leaf switches
```

Procedure

To edit the parameters in the configuration file

1. Open the `ciscoaci_observer_common.sh` configuration file and edit (at least) the following parameters:

   `ciscoapic_api_url`
   Cisco APIC REST API endpoint

ciscoapic_username
Cisco APIC user name for REST API

ciscoapic_password
Cisco APIC user password for REST API.

Supply the Cisco APIC user password in encrypted text.

ciscoapic_tenant_name
Cisco APIC tenant

Set to 'admin' if there is no specific tenant

Set to '' to load Fabric Topology resources

ssl_truststore_file
Cisco APIC SSL trust store file for HTTPS authentication

JKS is the supported format and the file is relative to `$ASM_HOME/security`

password_ssl_truststore_file
Password to decrypt and encrypt Cisco APIC SSL trust store file.

Supply Cisco APIC SSL trust store password in encrypted text.

Encryption requirement:

The Load and Listener jobs require passwords in encrypted form. To encrypt the `ciscoapic_password` and `password_ssl_truststore_file`, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

To acquire a Cisco APIC SSL certificate and build the SSL truststore

2. Use the following command to use OpenSSL to connect to Cisco APIC over port 443, and extract a SSL Certificate from Cisco APIC to a *<certificate_file_name>*.crt file.

```
echo -n | openssl s_client -connect {Cisco APIC IpAddress}:443 | sed -ne
'/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
```

3. Use the following Java keytool command to import the Cisco APIC certificate file into a keystore and encrypt the keystore with a given password.

```
keytool -import -v -trustcacerts -alias {Cisco APIC Hostname}
-file {certificate_file_name}.crt -keystore {keystore file name}
-storepass {your plain text password to encrypt keystore}
```

Tip: You will need the following encryption information when editing `ciscoaci_observer_common.sh`

Table 39. Encryption parameters required for `ciscoaci_observer_common.sh`

keystore parameter	ciscoaci_observer_common.sh parameter
keystore password	password_ssl_truststore_file
keystore file name	ssl_truststore_file

4. Copy the keystore file ({keystore file name}) to the `$ASM_HOME/security` directory to complete the SSL setup.

To start the Load and Listener jobs

5. To start the Cisco ACI Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/ciscoaci_observer_query_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need Cisco ACI topology data refreshed.
6. To start the Cisco ACI Observer Listener job, use the following command:

   ```
   $ASM_HOME/bin/ciscoaci_observer_listen_start.sh
   ```

   This job monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

What to do next

You can also use the following scripts:

ciscoaci_observer_query_stop.sh
: Stops the Full Topology Upload job

ciscoaci_observer_listen_stop.sh
: Stops the Listener job

ciscoaci_observer_job_list.sh
: Lists the status of current jobs

ciscoaci_observer_log_level.sh
: Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Contrail Observer jobs

The Contrail observer is installed as part of the core installation procedure. Using the Contrail Observer, you can retrieve topology data from Juniper Network Contrail Release 4.1 via REST APIs exposed by the Contrail API server. This observer is developed against Juniper Network Contrail that integrates with OpenStack orchestration platform (Ubuntu 16.04 + Contrail Cloud - Ocata).

Before you begin

Ensure you have the Contrail service details to hand, such as the username, password, and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/contrail-observer/swagger`

About this task

Contrail Observer jobs retrieve topology data from Juniper Network Contrail Release 4.1 via REST APIs exposed by the Contrail API server. The observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

contrail_observer_common.sh
The configuration file you use to customize Contrail Observer settings.

The parameters defined here are then used by the `contrail_observer_load_start.sh` and `contrail_observer_listen_start.sh` scripts to trigger the Contrail Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following jobs. You must edit the parameters in the configuration file before running these jobs.

Load job
A transient (one-off) job that loads all requested topology data.

This job is started by the `contrail_observer_load_start.sh` script and loads all supported resources.

Run this job whenever you need the Contrail topology data refreshed.

Listener
A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

This job is started by the `contrail_observer_listen_start.sh` script and loads all supported resources during startup, and listens to RabbitMQ messages from 'vnc_config.object-update' fanout exchange.

There is no need to run the Load job before running the Listen job, because the Listen job performs a Load job during initialization.

Table 40. Mapping of Contrail object types to Agile Service Manager entity types:

Contrail object types	Agile Service Manager entity types
domain	domain
project	project
bgp-as-a-service	service
bgpvpn	vpn
loadbalancer	loadbalancer
logical-router	router
network-ipam	ipam
service-instance	service
virtual-ip	ipaddress
virtual-machine-interface	networkinterface
virtual-network	network
virtual-router	router
physical-router	router
global-system-config	group
instance-ip	ipaddress
routing-instance	vrf
bgp-router	router

Table 40. Mapping of Contrail object types to Agile Service Manager entity types: (continued)

Contrail object types	Agile Service Manager entity types
route-target	routetarget

Procedure

To edit the parameters in the configuration file

1. Open the `contrail_observer_common.sh` configuration file and edit (at least) the following parameters:

   api_server_url
   Contrail API URL on which the Contrail API server is running

   os_auth_url
   Openstack authentication URL for the Identity service

   os_user
   Openstack username

   os_password
   Openstack password, in encrypted form

   Encryption requirement:

   The Load and Listener jobs require the Contrail token in encrypted form. To encrypt the token, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

2. Still the `contrail_observer_common.sh` configuration file, edit (at least) the following Listen parameters:

   rabbit_server
   Hostname or IP address of RabbitMQ server

   rabbit_user
   The username to authenticate with RabbitMQ

   rabbit_password
   The encrypted password to use authenticate with RabbitMQ

   Encryption requirement:

   The Load and Listener jobs require the Contrail token in encrypted form. To encrypt the token, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

To start the Load and Listener jobs

3. To start the Contrail Observer Load job, use the following command:

   ```
   $ASM_HOME/bin/contrail_observer_load_start.sh
   ```

This job loads all requested topology data. This job runs only once.

4. To start the Contrail Observer Listener job, use the following command:

   ```
   $ASM_HOME/bin/contrail_observer_listen_start.sh
   ```

   This job monitors its source for updates and runs until it is explicitly stopped, or until the observer is stopped.

What to do next

You can also use the following scripts:

contrail_observer_load_stop.sh
: Stops the Load job

contrail_observer_listen_stop.sh
: Stops the Listener job

contrail_observer_job_list.sh
: Lists the status of current jobs

contrail_observer_log_level.sh
: Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining DNS Observer jobs

The DNS Observer is installed as part of the core installation procedure. Using the DNS Observer, you can query internal DNS server performance, and use the returned information on response times and service addresses to create topologies within the topology service. The DNS Observer supports forward and reverse lookup calls, with **recurse** or **no recurse** options.

Before you begin

Ensure you have the DNS access details to hand, such as domain, DNS server address and port number.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/dns-observer/swagger`

About this task

The DNS Observer (nasm-dns-observer) provides DNS query services and topological insight into how a specified DNS server is performing forward (name-to-IP address) or reverse (IP address-to-name) lookups. Query results include a list of addresses, information on how long it takes the DNS server to resolve a lookup, and, optionally (with the maximum number of recursive calls set at 200) how the DNS server is recursively resolving a given name or IP address.

Job data is automatically posted to the topology service, after which the job status expires, after a set amount of time. The Topology Viewer displays the results with

color-coded lines representing the relationships between resources, and the lookup time in ms. A tabular view of the relationship details is also available.

Tip: The relationship types can be customized with line color, width and pattern functions. See the "Creating custom relationship type styles" on page 202 topic for more information.

dns_observer_common.sh
The configuration file you use to customize DNS Observer settings.

The parameters defined here are then used by the DNS forward and reverse lookup scripts (`dns_observer_forward_lookup_start.sh` and `dns_observer_reverse_lookup_start.sh`) to trigger the DNS Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following jobs. You must edit the parameters in the configuration file before running these jobs.

Forward lookup job
A transient (one-off) job that loads all requested DNS forward lookup topology data.

This job is started by the `dns_observer_forward_lookup_start.sh` script.

Reverse lookup job
A transient (one-off) job that loads all requested DNS reverse lookup topology data.

This job is started by the `dns_observer_reverse_lookup_start.sh` script.

Procedure

1. Open the `dns_observer_common.sh` configuration file and edit the required parameters.

   type Values can be either `forward` for name-to-IP address lookups, or reverse for IP address-to-name lookups.

   address_type
   IPV4 or IPV6

   server The DNS server IP address

   port The DNS server port number

   recurse
   Values can be `false` to run the job without recursion, or `true` to initiate recursion, with the maximum number of calls set at 200.

   domain_name
   The domain name for the DNS forward lookup job

   ip_address
   The IP address for the DNS reverse lookup job

Run the jobs

2. To start the DNS forward lookup job, use the following command:
   ```
   $ASM_HOME/bin/dns_observer_forward_lookup_start.sh
   ```
3. To start the DNS reverse lookup job, use the following command:

```
$ASM_HOME/bin/dns_observer_reverse_lookup_start.sh
```

Results

Data retrieved from the DNS query is displayed in the Agile Service Manager Topology Viewer.

Example

Example of a **forward** DNS Observer job **with no** recursive lookup:

```
{
  "unique_id": "my job",
  "type": "forward",
  "parameters": {
    "address_types": "IPV4",
    "server": "8.8.8.8",
    "port": 53,
    "recurse": false,
    "domain_name": "yourdomain.com"
  }
}
```

Example of a **reverse** DNS Observer job **with** recursive lookup:

```
{
  "unique_id": "my job",
  "type": "reverse",
  "parameters": {
    "address_types": "IPV4",
    "server": "8.8.8.8",
    "port": 53,
    "recurse": true,
    "ip_address": "8.8.8.8"
  }
}
```

What to do next

You can also use the following scripts:

dns_observer_lookup_stop.sh
: Stops the DNS observer lookup job

dns_observer_job_list.sh
: Lists the status of current jobs

dns_observer_log_level.sh
: Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Docker Observer jobs

Using the Docker Observer functionality, you can discover Docker network resources, including Docker Swarm clusters, and then visualize this data as a topology view in the Agile Service Manager UI. In addition, it also discovers Docker clusters managed by Docker UCP.

Before you begin

The observer supports Docker UCP v3.1.0.

Note: Docker UCP v3.1.0 supports only TLS 1.2 for SSL negotiation and has removed support for TLS 1 and TLS 1.1.

The Docker Observer is installed as part of the core installation procedure.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/docker-observer/swagger`

About this task

The Docker Observer performs a single load job, which performs a one-off discovery of the Docker network.

The job definition indicates whether to connect to a local Docker on the same (UNIX) host as the observer using the **`unix_socket`** parameter, or to a remote Docker using the **`host`** and **`port`** parameters.

Local docker

The default, if the job parameters are empty, is to try to connect to a UNIX socket at `/var/run/docker.sock`

If the location of the UNIX socket differs, the full path can be given in the **`unix_socket`** parameter. The **`host`** and **`port`** parameters must **not** be supplied.

In either case, the socket must be accessible.

When the observer is running within the docker container to be monitored, `/var/run/docker.sock` must be available within the container. For example:

```
volumes:
/var/run/docker.sock:/var/run/docker.sock
```

Remote docker

The host and port parameters of the job can be used to identify the TCP port that Docker can be reached on. The **`unix_socket`** parameter must **not** be supplied.

Docker is **not** accessible via TCP by default. To enable it, edit the `docker.service` file. On RedHat, this is available in `/usr/lib/systemd/system`. Amend the **`ExecStart`** option under the Service section to include a `-H` option. For example, to make it available externally on port 2375, you could add `-H tcp://0.0.0.0:2375`.

Note: If you want to continue to be able to access Docker via the default socket, for example if the Docker Observer container needs access, or if you want to be able to perform `docker ps -a` rather than `docker -H tcp://0.0.0.0:2375 ps -a`, then you need to also list it in the same line, as on the following example:

```
-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock
```

You must reload the configuration:

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Tip: If this fails to start Docker, and a Unix socket (or no socket at all) was specified, check that no directory with that name exists. If you start up docker with just a TCP socket and no Unix socket, this creates a `/var/run/docker.sock` directory, which you must delete after Docker is stopped, so that you can restart with access via that Unix socket.

`docker_observer_common.sh`

The configuration file you use to customize Docker Observer settings.

The parameters defined here are then used by the `docker_observer_load_start.sh` script to trigger the Docker Observer job.

You can use the **`view_all`** and **`exclude_containers`** parameters to filter the scope of observations. These parameters are arrays or lists that can accept multiple values.

view_all

Use this parameter to force modeling of all containers, tasks and images.

By default, only running containers, running tasks, and images currently in use by modeled containers are modeled.

exclude_containers

Use this parameter to filter out containers that are not of interest, based on regular expression matches against the container name.

Swagger UI usage examples

Using the Docker Observer, you can discover the following Docker resources:

- Remote Docker network resources via HTTP through TCP port exposure. Example:

```
{
  "unique_id": "my job",
  "type": "load",
  "parameters": {
    "host": "1.2.3.4",
    "port": 2375
  }
}
```

- Remote Docker network resources with HTTPS using a certificate. Example:

```
{
  "unique_id": "my job",
  "type": "load",
  "parameters": {
    "host": "1.2.3.4",
    "port": 2375,
    "username": "username",
    "password": "password",
    "docker_ssl_certificate": "certificate_file_name.crt",
    "docker_ssl_truststore_file": "truststore_file_name.jks",
    "password_ssl_truststore_file": "truststore_password"
  }
}
```

- Remote Docker network resources with HTTPS using certificate and truststore. Example:

```
{
  "unique_id": "my job",
  "type": "load",
  "parameters": {
    "host": "1.2.3.4",
    "port": 2375,
    "username": "username",
    "password": "password",
    "docker_ssl_certificate": "certificate_file_name.crt",
    "docker_ssl_truststore_file": "truststore_file_name.jks",
    "password_ssl_truststore_file": "truststore_password"
  }
}
```

- Remote Docker network resources with HTTPS using truststore. Example:

```
{
  "unique_id": "my job",
  "type": "load",
  "parameters": {
    "host": "1.2.3.4",
    "port": 2375,
    "username": "username",
    "password": "password",
    "docker_ssl_truststore_file": "truststore_file_name.jks",
    "password_ssl_truststore_file": "truststore_password"
  }
}
```

Procedure

1. Edit the `docker_observer_common.sh` config file as required. The Docker Observer supports multiple types of Docker configurations. Edit or populate the following parameters for different docker configurations:

   Local Docker
   Empty parameters

   Remote Docker with HTTP
   Populate **`host`** and **`port`**

   Remote Docker with HTTPS via certificate
   The certificate will be added to the named truststore

   Populate **`host`**, **`port`**, **`username`**, **`password`**, **`docker_ssl_certificate`**, **`docker_ssl_truststore_file`** and **`password_ssl_truststore_file`**

   Remote Docker with HTTPS via truststore
   The truststore must contain the certificate

   Populate **`host`**, **`port`**, **`username`**, **`password`**, **`docker_ssl_truststore_file`** and **`password_ssl_truststore_file`**

   Encryption requirement:

   All jobs require passwords in encrypted form. To encrypt 'password' and 'password_ssl_truststore_file', run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

2. To start a Docker Observer **Load** job, use the following command:
   ```
   $ASM_HOME/bin/docker_observer_load_start.sh
   ```

   Usage examples for starting jobs:

   Default job
   ```
   $ASM_HOME/bin/docker_observer_load_start.sh
   ```

   Local Docker
   ```
   env unique_id=My job name $ASM_HOME/bin/docker_observer_load_start.sh
   ```

   Remote Docker with HTTP
   ```
   env unique_id=My job name host=1.2.3.4 port=2375
   $ASM_HOME/bin/docker_observer_load_start.sh
   ```

   Remote Docker with HTTPS via certificate
   ```
   env unique_id=My job name host=1.2.3.4 port=2375 username=username
   password=password docker_ssl_certificate=certificate_file_name.crt
   docker_ssl_truststore_file=truststore_file_name.jks
   password_ssl_truststore_file=truststore_password
   $ASM_HOME/bin/docker_observer_load_start.sh
   ```

   Remote Docker with HTTPS via truststore
   ```
   env unique_id=My job name host=1.2.3.4 port=2375 username=username
   password=password docker_ssl_truststore_file=truststore_file_name.jks
   password_ssl_truststore_file=truststore_password $ASM_HOME/bin/
   docker_observer_load_start.sh
   ```

Results

The script triggers the Docker Observer Load job, which performs a one-off discovery of the Docker network you have specified.

What to do next

You can also use the following scripts:

docker_observer_load_stop.sh
: Stops the job

docker_observer_job_list.sh
: Lists the current job status

docker_observer_log_level.sh
: Sets the log level

Tip: For regular status updates, run the default Docker Observer job via a cron job.

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Dynatrace Observer jobs

The Dynatrace Observer is installed as part of the core installation procedure. Using the Dynatrace Observer, you can define jobs that dynamically load Dynatrace data for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the Dynatrace service details to hand, such as API token and Base URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/dynatrace-observer/swagger`

About this task

The Dynatrace Observer jobs extract Dynatrace resources via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`dynatrace_observer_common.sh`
: The configuration file you use to customize Dynatrace Observer settings.

  The parameters defined here are then used by the `dynatrace_observer_load_start.sh` to trigger the Dynatrace Observer jobs.

  Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
: A transient (one-off) job that loads all requested topology data.

  This job is started by the `dynatrace_observer_load_start.sh` script.

Procedure

To edit the parameters in the configuration file

1. Open the `dynatrace_observer_common.sh` configuration file and edit (at least) the following Load parameters:

   `api_token`
   : Dynatrace API token

   `base_url`
   : Dynatrace API base URL

   Encryption requirement:

   The Load job requires the API token in the configuration file in encrypted form. To encrypt the **`api_token`**, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

Enter and then confirm the API token. The encryption utility will return an encrypted **api_token**.

To start the Load job

2. To start the Dynatrace Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/dynatrace_observer_load_start.sh
   ```

Results

This job loads all requested topology data, and runs only once. Run this job whenever you need Dynatrace topology data refreshed.

What to do next

You can also use the following scripts:

dynatrace_observer_load_stop.sh
Stops the Load job

dynatrace_observer_job_list.sh
Lists the status of current jobs

dynatrace_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Event Observer jobs

You use the Event Observer to get events from Netcool/OMNIbus, via the XML Gateway, into the Netcool Agile Service Manager topology service. Netcool/OMNIbus events can also be generated from Agile Service Manager status via the Netcool/OMNIbus Message Bus probe. The Event Observer is installed as part of the core installation procedure.

Before you begin

Before you define an Event Observer job, you must configure the Netcool/OMNIbus XML Gateway and the Netcool/OMNIbus Message Bus probe, as described in the following topics:

- "Deploying the XML Gateway for Event Observer" on page 23
- "Deploying the Netcool/OMNIbus probe for Message Bus" on page 30

The Event Observer requires:

- Netcool/OMNIbus Probe for Message Bus **Version 8** or later
- Netcool/OMNIbus XML Gateway for Message Bus **Version 9** or later

Remember: Most prerequisites are deployed during the Agile Service Manager core installation. This **excludes** the Netcool/OMNIbus XML Gateway and the Netcool/OMNIbus Message Bus probe, which you must download, install and configure separately, but **includes** the Event Observer docker container, which has

been installed and should be running, as well as the required scripts to manage jobs. All observers have scripts to start and stop all available jobs, to list the status of a current job, to set its logging levels, and to configure its job parameters.

Note: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/event-observer/swagger`

About this task

The Event Observer runs a single long-running job for each tenant. This job listens for Netcool/OMNIbus events, which it receives via the Netcool/OMNIbus XML Gateway, and then sets the status of resources in the topology service.

The Event Observer receives batches of Netcool/OMNIbus events from the XML gateway. If it finds any matching resources in the Agile Service Manager topology service, it sets their status according to this event data, which you can then visualize in the Agile Service Manager UI.

`event_observer_common.sh`
The config file you use to customize Event Observer settings.

The parameters defined here are then used by the `event_observer_listen_start.sh` script to trigger the Event Observer job.

Table 41. Event Observer job parameters

Parameter	Action	Details
Unique ID	Enter a unique name for the job	Required
Netcool Sources	Specify the objects servers from which to obtain data.	Required (comma-separated list). To accept input from all object servers, set to '*' (asterisk).
Field names	Enter a (comma-separated) list of Netcool/OMNIbus ObjectServer `alerts.status` field names to identify top-level resources.	Optional. If empty, defaults to `"Node, NodeAlias"`. If **not** empty, all alerts.status field names must be listed. **Tip:** You can define extra event properties to be added to the status displayed in the Topology Viewer using the `extra_status_fields` property. Add a list of names, such as 'sourceId' or 'name'. You can then define topology viewer status tools that reference these.
Thread limit	Enter the number of received events to be processed in parallel.	Optional. The default is 100.
Observer job description	Enter additional information to describe the job.	Optional

Multi-tenant deployment

If you are connecting to more than one Netcool/OMNIbus ObjectServer, you can filter events received so that only events received from a specific ObjectServer are processed.

You specify an ObjectServer by editing the **netcool_sources** parameter in the `event_observer_common.sh` config file.

For example, you can change the following default parameter, which accepts all events (and is set to process 100 received events in parallel):

```
"parameters": {
    "netcool_sources": "*",
    "thread_limit": 100
   }
```

You can change it to the following, which accepts events only from the ObjectServer named `NCOMS_A` (and is set to process 90 received events in parallel):

```
"parameters": {
    "netcool_sources": "NCOMS_A",
    "thread_limit": 90
   }
```

Procedure

1. Edit the `event_observer_common.sh` config file as required.
2. To start the Event Observer **Listener** job, use the following command:

   ```
   $ASM_HOME/bin/event_observer_listen_start.sh
   ```

   Remember: The start script triggers the job, but is not the job itself.

Results

The Listener job monitors the Netcool/OMNIbus XML Gateway for updates and runs until it is stopped, or until the Observer is stopped.

What to do next

You can also use the following scripts:

event_observer_listen_stop.sh
: Stops the job

event_observer_job_list.sh
: Lists the current job status

event_observer_log_level.sh
: Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining File Observer jobs

Using the File Observer functionality, you can write bespoke data to a file in a specific format, upload this data to the topology service, and then visualize this data as a topology view in the Agile Service Manager UI. The File Observer is installed as part of the core installation procedure.

Before you begin

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/file-observer/swagger`

About this task

File Observer jobs are HTTP POST requests that can be triggered via cURL or swagger, or via the example scripts provided in the `$ASM_HOME/bin` directory.

`file_observer_common.sh`
: The config file you use to customize the File Observer job unique_id or service host.

  The parameters defined here are then used by the `file_observer_load_start.sh` script to trigger the File Observer job.

The File Observer runs a 'loadFile' job that loads all requested topology data for each tenant. The loadFile job takes the name of the file to parse and load.

Lines starting with `V:` (vertex), `E:` (edge), `D:` (delete) or `W:` (wait) are treated as instruction lines to be processed. Other lines, for example lines that are empty or commented out, are ignored.

Line format

V: The JSON payload takes the format described in the swagger documentation of the `POST /resources` message body.

E: The JSON payload takes the format described in the swagger documentation for the `_references` section of the `POST /resources` message body.

W: Takes an integer period followed by a string specifying the units.

D: Takes a single string which is the unique ID of the vertex to delete.

Tip: An example file is available in the `$ASM_HOME/data/file-observer` directory.

Restriction:

Files to be read by File Observer must be located in the following directory: `$ASM_HOME/data/file-observer`

A file name specified in a File Observer job must be relative to that directory (and not absolute).

Procedure

1. Edit the `file_observer_common.sh` config file.
2. Define your data file and copy the file to the following location: `/opt/ibm/netcool/asm/data/file-observer/` For example:

   ```
   cp dncim.file $ASM_HOME/data/file-observer/
   ```

3. To start the File Observer **Load** job, use one of the following commands:

   To define the data file via a command line argument

   ```
   ./file_observer_load_start.sh --file dncim.file
   ```

   To define the data file via the environment

   ```
   env file=dncim.file $ASM_HOME/bin/file_observer_load_start.sh
   ```

   The load job loads all requested topology data from the file specified. This job runs only once.

Example

The following cURL command example invokes the File Observer job:

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' -
d '{ "unique_id":
"dncim.file", "type": "load", "parameters": {   "file": "dncim.file"  } }'
http://localhost/1.0/file-observer/jobs
```

What to do next

You can also use the following scripts:

file_observer_load_stop.sh
: Stops the job

file_observer_job_list.sh
: Lists the current job status

file_observer_log_level.sh
: Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining IBM Cloud Observer jobs

The IBM Cloud Observer is installed as part of the core installation procedure. Use the IBM Cloud Observer when you have IBM Cloud installed in your environment to define jobs that perform REST calls to the IBM Cloud REST API. These jobs retrieve Cloud Foundry Apps information and services, and then dynamically load the retrieved data for analysis by Netcool Agile Service Manager.

Before you begin

Important: The IBM Cloud Observer supports Cloud Foundry API version 2.75.

Ensure you have the IBM Cloud access details to hand, such as username, password and region.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/ibmcloud-observer/swagger`

About this task

In a typical IBM Cloud environment, you have access to four different region:

- US_S (Dallas)
- UK (London)
- EU (Frankfurt)
- AP (Sydney & Tokyo)

You define which region is to be discovered, as IBM Cloud Observer supports all four regions.

Each region has its own URI, thus only a single region is discovered in a full load job. To discover different regions, a full load job needs to be triggered for each region. The prerequisites for a full load job are the IBM Cloud username, password and region.

Note: No listening job is supported at the moment.

Tip: You can configure IBM Cloud resources via the IBM Cloud GUI or the Cloud Foundry CLI.

`ibmcloud_observer_common.sh`
The config file you use to customize IBM Cloud Observer settings.

The parameters defined here are then used by the `ibmcloud_observer_load_start.sh` script to trigger the IBM Cloud Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

You define and start the following job. You must edit the parameters in the config file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `ibmcloud_observer_load_start.sh` script.

Table 42. Mapping IBM Cloud model objects to Agile Service Manager entity types

IBM Cloud resource object	Agile Service Manager entity types
stacks	operatingsystem
apps	application
routes	path
service bindings	hub
service instance	service
user provided service instance	service
spaces	group
organization	organization
buildpacks	component

Procedure

To edit the parameters in the config file

1. Open the `ibmcloud_observer_common.sh` config file and edit (at least) the following parameters:

   `username`
   The user name for the IBM Cloud REST API

   `password`
   The user password for the IBM Cloud REST API

   Encryption requirement: Jobs require the password in the configuration file to be encrypted. You encrypt the password using the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

   `region` The IBM Cloud resource region (supported region codes are US_S, UK, EU or AP)

To start the Load job

2. To start the IBM Cloud Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/ibmcloud_observer_load_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need the IBM Cloud topology data refreshed.

What to do next

You can also use the following scripts:

ibmcloud_observer_load_stop.sh
Stops the Full Topology Upload job

ibmcloud_observer_job_list.sh
Lists the status of current jobs

ibmcloud_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **`-h`** or **`--help`** to display help information, and with **`-v`** or **`--verbose`** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Kubernetes Observer jobs

The Kubernetes Observer is installed as part of the core installation procedure. Using this observer, you can define jobs that discover the services you run on Kubernetes, and display Kubernetes containers and the relationships between them.

Before you begin

Ensure you have the Kubernetes service details to hand, such as the Kubernetes host IP and SSL Certificate details.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/kubernetes-observer/swagger`

About this task

The Kubernetes Observer jobs query Kubernetes and extract information. The Observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

`kubernetes_observer_common.sh`
: The configuration file you use to customize Kubernetes Observer settings.

  The parameters defined here are then used by the `kubernetes_observer_query_start.sh`, the `kubernetes_observer_poll_start.sh` and the `kubernetes_observer_listen_start.sh` scripts to trigger the Kubernetes Observer jobs.

You must edit the parameters in the configuration file before running these jobs.

Load A transient (one-off) job that loads all requested topology data.

This job is started by the `kubernetes_observer_query_start.sh` script.

Poll A job that loads all requested topology data like the Load job, but repeated at set polling intervals.

This job is started by the `kubernetes_observer_poll_start.sh` script.

Weave Scope Listen
: A standalone job that listens to the Weave Scope agent and continues to stream topology and state data to Agile Service Manager.

  The listen job can maximally provide visibility of your Kubernetes services, pods, containers, deployments, stateful sets, Cron Jobs and processes for a specified namespace.

  A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

  This job is started by the `kubernetes_observer_listen_start.sh` script.

What to do next

Tip: You can start a job without editing the `kubernetes_observer_common.sh` script by providing a Kubernetes host IP or encrypted token directly, as in the following examples:

```
env kubernetes_token=<eyJhbGciOiJSUzI1NiIsInR5cCI6Ik>
/opt/ibm/netcool/asm/bin/kubernetes_observer_query_start.sh
```

```
env kubernetes_master_ip=<host ip>
/opt/ibm/netcool/asm/bin/kubernetes_observer_query_start.sh
```

You can also use the following scripts:

kubernetes_observer_query_stop.sh
Stops the Load job

kubernetes_observer_poll_stop.sh
Stops the Poll job

kubernetes_observer_listen_stop.sh
Stops the Weave Scope Listener job

kubernetes_observer_job_list.sh
Lists the status of current jobs

kubernetes_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

To define the full load and poll jobs

Before you begin

Required: Before defining a Kubernetes Observer Load or Poll job, you must create a service account in the Kubernetes environment and obtain its token.

1. Create a configuration file called `asm-k8s-observer.yaml` with the custom cluster role `asm:kubernetes-observer`

   Use the following sample content

   ```
   kind: ClusterRole
   apiVersion: rbac.authorization.k8s.io/v1
   metadata:
     annotations:
       rbac.authorization.kubernetes.io/autoupdate: "true"
     name: asm:kubernetes-observer
   rules:
   - apiGroups: ["", "extensions"]
     resources: ["replicasets", "pods", "events", "namespaces", "nodes", "services",
   "deployments"]
     verbs: ["get", "list", "watch"]
   ```

   Run the following command to create the `asm:kubernetes-observer` custom cluster role with 'read' access to the resources that the observer discovers, for example pods, namespaces, and nodes.

   ```
   kubectl create -f asm-k8s-observer.yaml
   ```

   Tip: Verify that the cluster role `asm:kubernetes-observer` and its privileges exist using the following commands:

   ```
   kubectl get clusterrole asm:kubernetes-observer
   ```

   ```
   kubectl describe clusterrole asm:kubernetes-observer
   ```

2. Create a service account:

   ```
   kubectl create serviceaccount asm-k8s-account
   ```

   Tip: Verify that the service account exists:

```
kubectl get serviceaccount
```

3. Bind the `asm:kubernetes-observer` role to the `asm-k8s-account` service account.

```
kubectl create clusterrolebinding asm-k8s --clusterrole=asm:kubernetes-observer
--serviceaccount=default:asm-k8s-account
```

4. Obtain the Kubernetes service account token by completing the following steps:
   a. Get all secrets:

```
kubectl get secret
```

   b. Describe the `asm-k8s-account-token-*******` (which in this example is *ch47f*):

```
kubectl describe secret asm-k8s-account-token-ch47f
```

Procedure

To edit the parameters in the `kubernetes_observer_common.sh` configuration file

1. Open the `kubernetes_observer_common.sh` configuration file and edit the following parameters:

data_center
Data centre running the Kubernetes instance, for example `dataCenter1`.

This parameter is used to ensure that observations of different Kubernetes clusters do not clash, and can be any string except `icp`, which is reserved for the ICP health view.

kubernetes_master_ip
Kubernetes host IP

kubernetes_token
Kubernetes service account token, which must be encrypted.

Encryption requirement: The Load job requires the token in the configuration file to be encrypted. You encrypt the Kubernetes token using the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

kubernetes_namespace
Optional parameter

Without the kubernetes_namespace parameter, the Kubernetes Observer uploads resources from all namespaces in the Kubernetes environment. With the parameter defined, the Kubernetes observer uploads resources only from the given namespace in the Kubernetes environment.

Tip: Run the following command in the Kubernetes environment to get a list of namespaces:

```
kubectl get namespaces
```

kubernetes_api_port
The Kubernetes API Port

Tip: Get the Kubernetes master IP and its API port using the following command:

```
kubectl cluster-info
```

The system returns the following information:

```
Kubernetes master is running at https://{master}:{port}
```

ssl_certificate_file

Kubernetes SSL certificate file name, which is the name of a file within `$ASM_HOME/security`. The files in the `$ASM_HOME/security` directory are made available in the observer container.

Tip: Obtain the SSL Certificate:

a. Get the kubernetes master IP and its API port using:

```
kubectl cluster-info
```

b. Run the following OpenSSL command:

```
echo -n | openssl s_client -connect {master ip}:{api} | sed -ne
'/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./certificate_file_name.crt
```

The certificate is saved as *certificate_file_name.crt*

c. Copy the certificate file to the `$ASM_HOME/security` directory.

connect_read_timeout_ms

Connection timeout in milliseconds (ms), for example '5000'.

hide_terminated_pods

The default value is 'false'.

- If you set this parameter to 'true', all pods with the phase `Succeeded` will be excluded from the Topology Viewer.
- If set to 'false', all pods regardless of any phase will be shown in Topology Viewer.

POLL_INTERVAL

You define the poll job POLL_INTERVAL in milliseconds to invoke full loading with the other provided values every *POLL_INTERVAL* milliseconds.

This value must be sufficiently large so that a new job is not submitted before the previous one has completed.

To start the Load job

2. To start the Kubernetes Observer **Load** job, use the following command:

```
$ASM_HOME/bin/kubernetes_observer_query_start.sh
```

The Load job loads all requested topology data. This job runs only once.

To start the Poll job

3. To start the Kubernetes Observer **Poll** job, use the following command:

```
$ASM_HOME/bin/kubernetes_observer_poll_start.sh
```

The Poll job loads all requested topology data at defined intervals.

To define the Weave Scope Listen job

Before you begin

Required: Before defining a Kubernetes Observer Weave Scope Listen job, you must install Weave Scope in your Kubernetes environment. For more information on Weave Scope, see the following location: https://www.weave.works/docs/scope/latest/introducing/

Update Note:

From Agile Service Manager 1.1.5 onwards, the location of the Weave Scope listen job changes. As a result, existing Weave Scope jobs that were running during an upgrade to Version 1.1.5 will have their paths renamed when the observer starts. However, Weave Scope jobs that were not running (stopped) will not be recognized and so will not have their path renamed. As a consequence, the UI will be unable to restart them.
Workaround: Ensure that your Weave Scope jobs are running before you update your system.

For IBM Cloud Private 2.1.0.3

1. Install Weave Scope as in the following example:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/scope.yaml?k8s-service-
   type=NodePort&k8s-version=$(kubectl version | base64 | tr -d '\n')"
   ```

   The installation creates a port (NodePort) that the Kubernetes Observer can use.
2. Identify the port using the following command:

   ```
   kubectl -n weave describe service weave-scope-app
   ```
3. Launch the Weave Scope User Interface:

   ```
   http://<master ip>:<NodePort>
   ```
4. If the UI is empty or you are experiencing connection issues, you check the pod and agent using the following options:

   `kubectl get -n weave pods`
   Gets all the pods from the weave namespace.

   The weave scope app pod should be running.

   `kubectl get -n weave daemonsets`
   Gets all daemonsets from the weave namespace.

   There should be a weave-scope-agent running per host in the Kubernetes cluster.

   `kubectl describe -n weave daemonsets weave-scope-agent`
   This command describes the weave scope agent daemonset.

   If the value for weave-scope-agent in the daemonsets is 0 (zero), a security error appears at the end in the events section.

   In the case of a security error, create the following configuration files:

   PodSecurityPolicy
   Example:

   ```
   apiVersion: extensions/v1beta1
   kind: PodSecurityPolicy
   metadata:
     name: weave-scope
   spec:
     privileged: true
   ```

```
  hostPID: true
  hostNetwork: true
  allowedCapabilities:
  - 'NET_ADMIN'
  fsGroup:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - '*'
```

ClusterRole

Example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: weave-scope
rules:
- apiGroups:
  - extensions
  resourceNames:
  - weave-scope
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

ClusterRoleBinding

Example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: weave-scope-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: weave-scope
subjects:
- kind: ServiceAccount
  name: weave-scope
  namespace: weave
```

kubectl apply -f <filename>

This command applies each of the configuration file.

On starting, the weave-scope-agent should now be ready and available.

For IBM Cloud Private 3.1.1

1. Create Namespace 'weave' with 'ibm-privileged-psp'.

   ```
   kubectl create namespace weave
   kubectl -n weave create rolebinding weave-clusterrole-rolebinding --
   clusterrole=ibm-privileged-clusterrole --group=system:serviceaccounts:
   weave
   ```

2. Install Weave Scope using the following command:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/scope.yaml?k8s-service-
   type=NodePort&k8s-version=$(kubectl version | base64 | tr -d '\n')"
   ```

   This will result in a port being opened that the Observer can use.

3. You can discover the NodePort using the following command:

```
kubectl -n weave describe service weave-scope-app
```

4. Launch the Weave Scope UI using the following URL:
   ```
   http://<master ip>:<NodePort>
   ```

Procedure

To edit the parameters in the `kubernetes_observer_common.sh` configuration file

1. Open the `kubernetes_observer_common.sh` configuration file and edit the following parameters:

   data_center
   Data centre running the Kubernetes instance, for example `mycluster`.

   kubernetes_master_ip
   Kubernetes host IP

   weavescope_port
   Weave Scope port, that is, `NodePort`.

   Tip: The NodePort can be obtained using the following command:
   ```
   kubectl -n weave describe service weave-scope-app
   ```

   namespace
   List of Kubernetes namespaces to listen for.

   Tip: Run the following command in the Kubernetes environment to get a list of namespaces:
   ```
   kubectl get namespaces
   ```

   topologies
   List of resources to include in the topology.

   Available resources:
   - containers
   - hosts
   - kube-controllers
   - pods
   - processes
   - services

   exclude_resources
   List of resources to exclude from the topology.

   The default is to exclude containers named 'POD' and kube-system resources.

To start the Weave Scope Listen job

2. To start the Kubernetes Observer **Weave Scope Listen** job, use the following command:
   ```
   $ASM_HOME/bin/kubernetes_observer_listen_start.sh
   ```

   The Listener job monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

What to do next

Tip: You can start a job without editing the `kubernetes_observer_common.sh` script by providing a Kubernetes host IP or encrypted token directly, as in the following examples::

```
env kubernetes_token=<eyJhbGciOiJSUzI1NiIsInR5cCI6Ik>

$ASM_HOME/bin/kubernetes_observer_query_start.sh
env kubernetes_master_ip=<host ip>

$ASM_HOME/bin/kubernetes_observer_query_start.sh
```

However, when setting any of these parameters in the file or on the command line, all parameters must be valid.

Defining Network Manager Observer jobs

The ITNM Observer is installed as part of the core installation procedure. Using the ITNM Observer, you can define jobs that dynamically load data discovered by IBM Tivoli Network Manager for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the ITNM service details to hand, such as the ITNM domain, host and port number.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/itnm-observer/swagger`

Update Note:

When updating from Agile Service Manager Version 1.1.3 to a later version, you must run a migration script to avoid the creation of duplicate ITNM Observer records **before** running any ITNM Observer jobs:

```
cd $ASM_HOME/bin
/execute_crawler.sh -c itnm_provider_transfer
```

- Running this script **before** making any new observations with the ITNM Observer prevents the creation of duplicate records.
- Running this script **after** making new observations with the ITNM Observer removes duplicate records, but **may not** preserve some historical topology data previously gathered by the ITNM Observer.

The script, which may take some time to complete on large topologies, creates a management artifact in the topology. You can monitor its progress by querying the artifact via Swagger.

About this task

The ITNM Observer jobs extract IBM Tivoli Network Manager resources using an Object Query Language JDBC driver. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`itnm_observer_common.sh`
: The config file you use to customize ITNM Observer settings.

The parameters defined here are then used by the `itnm_observer_load_start.sh` and the `itnm_observer_listen_start.sh` scripts to trigger the ITNM Observer jobs.

After installation, you define and start the following two jobs. You must edit the parameters in the config file before running these jobs.

Full Topology Upload
A transient (one-off) job that loads all requested topology data.

This job is started by the `itnm_observer_load_start.sh` script.

Listener
A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

This job is started by the `itnm_observer_listen_start.sh` script.

Procedure

1. Edit (at least) the following parameters in the `itnm_observer_common.sh` config file:

   `domain` Network Manager domain name

   `host` Network Manager server

   `port` Port used to access the Network Manager ncp_config process

   Note: The value of **`port`** will vary if multiple domains exist. To determine which port number to use for a Network Manager domain, look for the domain-specific ncp_config entry in the `$NCHOME/etc/precision/ServiceData.cfg` file.

   exclude_no_connection
   If true, only load entities that have connections including their dependencies are included.

   topology_type_edge_type_map
   Map of ITNM topology type to ASM edge/relationship type `{"topologyType":"edgeType"}` in JSON string format.

   The default value is `{"ConvergedTopology":"connectedTo"}`.

   The value of topology type can be found in `$NCHOME/precision/disco/stitchers/DNCIM/PopulateDNCIMTopologies.stch`

   Alternatively, run the following OQL statement against the model service to list the available topology type:

   ```
   select ENTITYNAME from ncimCache.entityData where METACLASS='Topology'
   ```

   Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

2. To start the ITNM Observer **Full Topology Upload** job, use the following command:

   ```
   $ASM_HOME/bin/itnm_observer_load_start.sh
   ```

   The Full Topology Upload job loads all requested topology data. This job runs only once.

3. To start the ITNM Observer **Listener** job, use the following command:

   ```
   $ASM_HOME/bin/itnm_observer_listen_start.sh
   ```

The Listener job monitors its source for updates and runs until it is stopped, or until the Observer is stopped.

What to do next

You can also use the following scripts:

itnm_observer_listen_stop.sh
Stops the Listener job

itnm_observer_load_stop.sh
Stops the Full Topology Upload job

itnm_observer_job_list.sh
Lists the current job status

itnm_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining New Relic Observer jobs

The New Relic Observer is installed as part of the core installation procedure. Use New Relic Observer when you have a New Relic account with a New Relic Infrastructure subscription. Using New Relic Observer, you can define jobs that dynamically load New Relic Infrastructure resource data via New Relic for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the New Relic account and New Relic Infrastructure subscription details to hand, such as the account name, account ID, and New Relic Insights API query key.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/newrelic-observer/swagger`

Restriction: New Relic applies a 1000 results limit on all New Relic Query Language (NRQL) queries. To accommodate this limit when retrieving data from the SystemSample, StorageSample, ProcessSample and NetworkSample event tables, the New Relic Observer uses the following NRQL query time clause:

```
"SINCE 4 hours ago LIMIT 1000"
```

About this task

The Observer uses the New Relic Infrastructure subscription and makes active New Relic Query Language (NRQL) calls over REST to New Relic Insights to download New Relic Infrastructure resource data.

The New Relic Observer loads the following New Relic Infrastructure resources and their relationships to the Agile Service Manager core topology service:

- Host
- Storage
- OS
- Network Interfaces
- Processes

The New Relic Observer job extracts New Relic Infrastructure resources from New Relic using New Relic Query Language (NRQL) over REST. The observer loads and updates the resources and their relationships within the Agile Service Manager core topology service.

newrelic_observer_common.sh
The configuration file you use to customize New Relic Observer settings.

The parameters defined here are then used by the `newrelic_observer_load_start.sh` script to trigger the New Relic Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `newrelic_observer_load_start.sh` script.

Procedure

To edit the parameters in the configuration file

1. Open the `newrelic_observer_common.sh` configuration file and edit (at least) the following Load parameters:

   accountName
   New Relic account name or tenant name

   accountId
   New Relic account ID.

   To obtain the account ID, first log into the New Relic login page: https://login.newrelic.com/login and then obtain the account ID from this URL:
   https://rpm.newrelic.com/accounts/%3CaccountId%3E

   insightsQueryAPIKey
   New Relic Insights Query API Key in encoded format.

   A new Relic user with a new Relic Infrastructure subscription is required to generate a new Relic Insights query API Key as outlined here: https://docs.newrelic.com/docs/insights/insights-api/get-data/query-insights-event-data-api

   Use the Agile Service Manager encryption tool to encode the New Relic Insights query API key before using it in job parameter.

   Encryption requirement:

The Load job requires the `insightsQueryAPIKey` in encrypted form. To encrypt the insightsQueryAPIKey, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

filterCriteria
Extend the result set returned to Agile Service Manager.

The default is 'SINCE 4 hours ago LIMIT 1000'.

For more information, see the documentation for New Relic Query Language.

To start the Load job

2. To start the New Relic Observer Full Topology Upload job, use the following command:

```
$ASM_HOME/bin/newrelic_observer_load_start.sh
```

Results

This job loads all requested topology data. Run this job whenever you need New Relic topology data refreshed.

What to do next

You can also use the following scripts:

newrelic_observer_load_stop.sh
Stops the Load job

newrelic_observer_job_list.sh
Lists the status of current jobs

newrelic_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining OpenStack Observer jobs

The OpenStack Observer is installed as part of the core installation procedure. Using the OpenStack Observer, you can define jobs that dynamically load OpenStack data for analysis by Netcool Agile Service Manager.

Before you begin

Ensure you have the OpenStack service details to hand, such as username, password, and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/openstack-observer/swagger`

About this task

The OpenStack Observer jobs extract OpenStack resources via REST or RabbitMQ. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`openstack_observer_common.sh`
The config file you use to customize OpenStack Observer settings.

The parameters defined here are then used by the `openstack_observer_query_start.sh` and the `openstack_observer_listen_start.sh` scripts to trigger the OpenStack Observer jobs.

You define and start the following two jobs. You must edit the parameters in the config file before running these jobs.

Full Topology Upload
A transient (one-off) job that loads all requested topology data.

This job is started by the `openstack_observer_query_start.sh` script.

Restriction: An OpenStack environment that has a list of endpoints whereby the heat-cfn service comes first before the heat service, will encounter a JSON parsing error in the logs due to a known issue in the openstack4j library. When this happens, the full load for the heat service will be skipped entirely. The other service will run as normal.

Listener
A long-running job that monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

This job is started by the `openstack_observer_listen_start.sh` script.

Restriction: Only one listening job should be listening to one queue (or sets of queues) at any one time. If you need to listen to multiple projects, then separate queues must be set up in OpenStack, with appropriate names, before separate listening jobs are submitted for each. For example, for Nova via the rmq_nova_notify attribute, for Neutron via the rmq_neutron_notify attribute.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

Procedure

To edit the parameters in the `openstack_observer_common.sh` config file

1. Open the `openstack_observer_common.sh` config file and edit (at least) the following **Load** parameters:

   `os_auth_url`
   OpenStack identity endpoint

   `os_username`
   OpenStack user name

os_password
OpenStack user password

Encryption requirement:

The Load and Listener jobs require passwords in the configuration file to be encrypted. To encrypt the os_password, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password, for example:

```
2IuExvqz5SGnGgROYGLAQg==
```

os_tenant_name
OpenStack tenant

Restriction: The **os_tenant_name** parameter should only be specified for jobs of version 2 authentication (and **not** version 3). When using authentication version 3, specify the **os_project_name** parameter in place of the **os_tenant_name** parameter.

os_perspective
OpenStack network perspective

2. Still in the `openstack_observer_common.sh` config file, edit (at least) the following **Listen** parameters:

rmq_hosts
RMQ connection details

rmq_username
RMQ user name

rmq_password
RMQ user password

Encryption requirement:

The Load and Listener jobs require passwords in the configuration file to be encrypted. To encrypt the rmq_password, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

os_project_name
OpenStack project

Remember: The **os_project_name** parameter should be specified in place of **os_tenant_name** when using authentication version 3.

To configure the OpenStack installation method

3. Do one of the following depending on whether you have used, or are planning to use, DevStack or another method to install OpenStack.
   - **DevStack installation**

If you have already installed OpenStack using DevStack
Add the following code to the end of the `local.conf` file, and then reinstall OpenStack.

If you are planning to install OpenStack using DevStack
Add the following code to the end of the `local.conf` file before installation.

```
[[post-config|$NOVA_CONF]]
[DEFAULT]
notification_topics = notifications,com.ibm.asm.obs.nova.notify
notification_driver=messagingv2
notify_on_state_change=vm_and_task_state
notify_on_any_change=True
```

- **Other installation**

  For standard (or any other) OpenStack installations
  Add the following code under the `[DEFAULT]` section of the `nova.conf` file, and then restart the nova compute service.

```
notification_topics = notifications,com.ibm.asm.obs.nova.notify
notification_driver=messagingv2
notify_on_state_change=vm_and_task_state
notify_on_any_change=True
```

To start the Load and Listener jobs

4. To start the OpenStack Observer **Full Topology Upload** job, use the following command:

   ```
   $ASM_HOME/bin/openstack_observer_query_start.sh
   ```

   The Full Topology Upload job loads all requested topology data. This job runs only once.

5. To start the OpenStack Observer listener job, use the following command:

   ```
   $ASM_HOME/bin/openstack_observer_listen_start.sh
   ```

   The Listener job monitors its source for updates and runs until it is explicitly stopped, or until the Observer is stopped.

What to do next

You can also use the following scripts:

openstack_observer_query_stop.sh
Stops the Full Topology Upload job

openstack_observer_listen_stop.sh
Stops the Listener job

openstack_observer_job_list.sh
Lists the status of current jobs

openstack_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **`-h`** or **`--help`** to display help information, and with **`-v`** or **`--verbose`** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining REST Observer jobs

The REST (or RESTful) Observer is installed as part of the core installation procedure. Use the REST Observer for obtaining topology data via REST endpoints. This observer is a counterpart to the File Observer.

Before you begin

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/rest-observer/swagger`

About this task

The REST Observer passes topology data to Agile Service Manager using a RESTful set of interfaces, which provide REST APIs that enable the following functionality:

- Management of Listen and bulk-replace job types.
- The insert-update (HTTP POST) of resources.
- The insert-update (HTTP POST) of relationships.
- The insert-replacement (HTTP PUT) of resources.
- The deletion (HTTP DELETE) of resources.
- A REST API that supports the deletion (HTTP DELETE) of all relationships of a given type from a specified resource.
- A REST API that supports the deletion (HTTP DELETE) of a specific relationship.

  Restriction: Resources created via REST can have a provider, but not an observer.

Benefits

Using the REST Observer rather than the File Observer or Topology Service APIs includes the following benefits:

- The ability to provide data to Agile Service Manager via HTTP REST Endpoints instead of files.
- The processing performed by all observers in their framework ensures that meta-data about observations from observers is managed correctly.
- A simple way of deleting all edges of a given type on a resource or a specific edge instance.

To use the REST Observer, a job request must be issued (HTTP POST) to the Observer instance job management APIs before sending data to the Resource and Relationship APIs.

Listen A long-running listen job capable of consuming topology data over a long period of time.

A listen job is designed to support scenarios where the input data stream is unpredictable, or there is little or no consistency or versioning of resources within the data stream.

Note: These examples assume that the environment variables have been set in `rest_observer_common.sh`

start

```
$ASM_HOME/bin/rest_observer_listen_start.sh
```

stop Default job

```
./bin/rest_observer_listen_stop.sh
```

Named job

```
env unique_id='My job name' $ASM_HOME/bin/rest_observer_listen_stop.sh
```

Bulk replace

A long-running job with the same resource replace semantics as the File Observer.

Bulk-replace jobs are designed to support scenarios where a known set of resources are subject to updates or versioning, and a prior observation about resources is to be replaced with a new one.

This job can provide a new set of resources and relationships and synchronize them to Agile Service Manager, thereby causing any previous data provided by the Observer to be deleted and replaced with the new data.

Note: These examples assume that the environment variables have been set in `rest_observer_common.sh`

start Default job:

```
./bin/rest_observer_bulk_replace_start.sh
```

Job with bulk_replace_unique_id and provider given:

```
env bulk_replace_unique_id=manageDataCenter provider=MyJavaProgram
$ASM_HOME/bin/rest_observer_bulk_replace_start.sh
```

synchronize

Default job

```
./bin/rest_observer_bulk_replace_synchronize.sh
```

Named job

```
env unique_id='My job name'
$ASM_HOME/bin/rest_observer_bulk_replace_synchronize.sh
```

stop Default job

```
$ASM_HOME/bin/rest_observer_bulk_replace_stop.sh
```

Named job

```
env unique_id='My job name'
$ASM_HOME/bin/rest_observer_bulk_replace_stop.sh
```

Once a job request has been successfully submitted, you can start to provide data to the Resource and Relationship APIs on behalf of a given job instance.

The Resource and Relationship APIs may respond with an `HTTP 503 Service Unavailable` response with a `Retry-After: 10 seconds` in the header. This indicates that even though the request against those APIs is valid, the observer has not been able to ascertain that meta-data about the job is held in Agile Service Manager yet; this may be due to, for example, any prevailing conditions in the network that support the Agile Service Manager micro-services.

Tip: If such a response is received, try the request again later.

Procedure

Listen job process and examples
The following procedure (steps one to ten) includes examples that show how to use the REST Observer listen job to create and adjust a small topology.

1. Start the Listen job. Use the following example as a model.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' -d '{
  "unique_id": "my job",
  "type": "listen",
  "parameters": {
    "provider": "MyListenJob"
  }
}' 'http://localhost/1.0/rest-observer/jobs/listen'
```

2. Verify that the job is running.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X GET --header 'Accept: application/json'
--header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
'http://localhost/1.0/rest-observer/jobs/my%20job'
```

3. Create a 'person' resource. This example creates a person resource called 'Thomas Watson'.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--header 'JobId: my job' -d '{
  "name": "Thomas Watson",
  "uniqueId": "Thomas Watson",
  "entityTypes": [
    "person"
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

4. Create an 'organization' resource. This example creates an 'organization' resource of 'IBM'.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job' -d '{
  "name": "IBM",
  "uniqueId": "IBM",
  "entityTypes": [
    "organization"
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

5. Create a 'manages' relationship between Thomas Watson and IBM.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job' -d '{
  "_fromUniqueId": "Thomas Watson",
  "_edgeType": "manages",
  "_toUniqueId": "IBM"
}' 'http://localhost/1.0/rest-observer/rest/references'
```

6. Create a new 'location' resource and relate it to Thomas Watson. This example creates a 'location' resource of 'Campbell, New York' for Thomas Watson, and a location relationship (an edgeType of locatedAt).

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job' -d '{
  "name": "Campbell, New York",
  "uniqueId": "Campbell, New York",
  "entityTypes": [
    "location"
  ],
  "_references": [
    {
      "_fromUniqueId": "Thomas Watson",
      "_edgeType": "locatedAt"
    }
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

7. Replace the location resource with one having latitude and longitude properties.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X PUT --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job' -d '{
  "name": "Campbell, New York",
  "uniqueId": "Campbell, New York",
  "entityTypes": [
    "location"
  ],
  "latitude": 42.232909,
  "longitude": -77.196918
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

8. Delete all locatedAt relationships from Thomas Watson.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X DELETE --header 'Accept: application/json'
--header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job'
'http://localhost/1.0/rest-observer/rest/resources/
Thomas%20Watson/references/both/locatedAt'
```

9. Delete the Campbell, New York location.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X DELETE --header 'Accept: application/json' --header
'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job'
'http://localhost/1.0/rest-observer/rest/resources/Campbell%2C%20New%20York'
```

10. Delete the manages relationship between Thomas Watson and IBM.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X DELETE --header 'Accept: application/json'
--header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my job'
'http://localhost/1.0/rest-observer/rest/resources/
Thomas%20Watson/references/both/manages/IBM'
```

Bulk Replace job process and examples

The following procedure (steps 11 - 21) includes examples that show how to use the REST Observer bulk-replace job to create and adjust a small topology.

Note: These examples use a mixture of the provided scripts in `$ASM_HOME/bin` and the cURL command.

11. Submit a bulk-replace job request with the unique ID of 'my bulk replace'.

```
[root@asm-backend asm]# env bulk_replace_unique_id="my bulk replace"
provider="Me" bin/rest_observer_bulk_replace_start.sh
```

12. Verify that the job is running.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X GET --header 'Accept:
application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
'http://localhost/1.0/rest-observer/jobs/my%20job'
```

13. Submit a location resource for the city of Coventry.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--header 'JobId: my bulk replace' -d '{
  "name": "Coventry",
  "uniqueId": "Coventry",
  "entityTypes": [
    "location"
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

14. Submit a location resource for the town of Rugby

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--header 'JobId: my bulk replace' -d '{
  "name": "Rugby",
  "uniqueId": "Rugby",
  "entityTypes": [
    "location"
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

15. Submit a location resource for the Town of Leamington Spa with relationships to the existing resources of Coventry and Rugby.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my bulk replace' -d '{
  "name": "Leamington Spa",
  "uniqueId": "Leamington Spa",
  "entityTypes": [
    "location"
```

```
  ],
  "_references": [
     {
        "_toUniqueId": "Coventry",
        "_edgeType": "routesVia"
     },
     {
        "_toUniqueId": "Rugby",
        "_edgeType": "routesVia"
     }
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

16. Check your progress by rendering the topology.
17. Having completed this set of observations, initiate a synchronize request for 'my bulk replace' job.

```
[root@asm-backend asm]# env bulk_replace_unique_id="my bulk
replace" ./bin/rest_observer_bulk_replace_synchronize.sh
```

18. Provide a new topology for the 'my bulk replace' job. This example submits a location resource for the town of Leamington Spa with relationships to the towns of Warwick and Stratford-Upon-Avon (resource placeholders).

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--header 'JobId: my bulk replace' -d '{
  "name": "Leamington Spa",
  "uniqueId": "Leamington Spa",
  "entityTypes": [
    "location"
  ],
  "_references": [
     {
        "_toUniqueId": "Warwick",
        "_edgeType": "routesVia"
     },
     {
        "_toUniqueId": "Stratford-upon-Avon",
        "_edgeType": "routesVia"
     }
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

19. Provide the resource data for the Town of Warwick resource placeholder, thus fully creating the resource.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type:
application/json' --header 'Accept: application/json' --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --header 'JobId: my bulk replace' -d '{
  "name": "Warwick",
  "uniqueId": "Warwick",
  "entityTypes": [
    "location"
  ]
}' 'http://localhost/1.0/rest-observer/rest/resources'
```

20. Provide the resource data for the town of Stratford-upon-Avon resource placeholder, thus fully creating the resource.

```
curl -u PROXY_USER[:PROXY_PASSWORD] -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--header 'JobId: my bulk replace' -d '{
  "name": "Stratford-upon-Avon",
  "uniqueId": "Stratford-upon-Avon",
  "entityTypes": [
    "location"
  ]
}' 'http://asm-backend.rtp.raleigh.ibm.com/1.0/rest-observer/rest/resources'
```

21. Initiate a synchronize request for the 'my bulk replace' job. This signifies to the Observer that it should instruct ASM to replace the previous set of observations with the new ones.

    Note: The new data is available immediately as it is provided to ASM. The synchronize request simply deletes any resources previously observed that were not observed this time. In the current example, Coventry and Rugby were not observed, and therefore they are deleted.

```
[root@asm-backend asm]# env bulk_replace_unique_id="my bulk
replace" ./bin/rest_observer_bulk_replace_synchronize.sh
```

What to do next

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining ServiceNow Observer jobs

The ServiceNow Observer is installed as part of the core installation procedure. Using the ServiceNow Observer, you can retrieve the configuration management database (CMDB) data from ServiceNow via REST API. Currently, the observer only supports load job. The load job queries the configuration items (CI) from CMDB via ServiceNow REST API using basic authentication credentials.

Before you begin

Ensure your user account has the `rest_api_explorer` and `web_service_admin` roles. These roles are required to access the resources from ServiceNow. Also, ensure you have the ServiceNow service details to hand, such as username, password, and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/servicenow-observer/swagger`

About this task

ServiceNow jobs retrieve the configuration management database (CMDB) data from ServiceNow via REST API. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`servicenow_observer_common.sh`
The configuration file you use to customize ServiceNow Observer settings.

The parameters defined here are then used by the `servicenow_observer_load_start.sh` script to trigger the ServiceNow Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the configuration file before running this job.

Load job
A transient (one-off) job that loads all requested topology data.

This job is started by the `servicenow_observer_load_start.sh` script.

Run this job whenever you need the ServiceNow topology data refreshed. .

Table 43. Mapping of ServiceNow object types to Agile Service Manager entity types:

ServiceNow object types	Agile Service Manager entity types
cmdb_ci	based on sys_class_name attribute
cmn_department	department
cmn_location	location
core_company	company
sys_user	person

Procedure

To edit the parameters in the configuration file

1. Open the `servicenow_observer_common.sh` configuration file and edit (at least) the following parameters:

   `ServiceNow username`
   The username of the ServiceNow instance

   `ServiceNow password`
   The password of the ServiceNow instance

   Encryption requirement:

   The Load job requires the password in the configuration file to be encrypted. To encrypt the password, run the `encrypt_password.sh` script in the `ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

   Instance URL
   The URL on which the ServiceNow instance is running

To start the Load job

2. To start the ServiceNow Observer Load job, use the following command:

   ```
   $ASM_HOME/bin/servicenow_observer_load_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need the ServiceNow topology data refreshed.

Results

You can now use data retrieved from the ServiceNow configuration management database (CMDB) to create topologies in the Agile Service Manager topology service.

What to do next

You can also use the following scripts:

servicenow_observer_load_stop.sh
Stops the Load job

servicenow_observer_job_list.sh
Lists the status of current jobs

servicenow_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining TADDM Observer jobs

The TADDM Observer is installed as part of the core installation procedure. Using the TADDM Observer, you can retrieve network topology data from the TADDM database and use this data to create topologies within the topology service.

Before you begin

Ensure you have the TADDM Rest API login access details in hand, such as the TADDM API URL, username and password.

All prerequisites are deployed during the Agile Service Manager core installation. This includes the TADDM Observer docker container, which has been installed and should be running, as well as the required scripts to manage jobs. All observers have scripts to start and stop all available jobs, to list the status of a current job, to set its logging levels, and to configure its job parameters.

You can verify that the TADDM Observer's docker container is running using the following command:

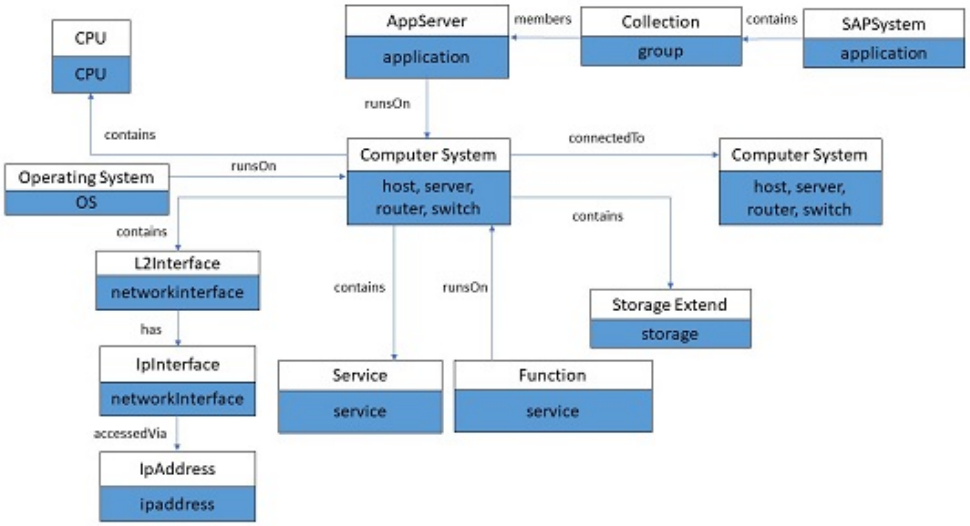
```
$ASM_HOME/bin/docker-compose ps
```

The system should return text indicating that `asm_taddm-observer_1` has a state of `Up` and therefore is running.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/taddm-observer/swagger`

About this task

The TADDM Observer is built on the Observer framework:



- A computer system can be a host, server, router or switch
- A computer system contains CPU, L2Interface and storage
- Operating system, application server and service run on a computer system
- A computer system can connect to another computer system
- A SAPSystem contains collection
- An application server can be a member of a collection

Table 44. Mapping TADDM model objects to Agile Service Manager entity types

TADDM model object	Agile Service Manager entity types
AppServer	application
ComputerSystem	host, server, router, switch
CPU	cpu
L2Interface	networkinterface
IpInterface	networkinterface
IpAddress	ipaddress
OperatingSystem	os
Service	service
StorageExtent	storage
Function	service
SAPSystem	application
Collection	group

The TADDM Observer job retrieves topology data using the TADDM REST API. The observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

taddm_observer_common.sh
: The config file you use to customize TADDM Observer settings.

The parameters defined here are then used by the `taddm_observer_load_start.sh` and `taddm_observer_poll_start.sh` scripts to trigger the TADDM Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the configuration file settings.

You define and start the following job. You must edit the parameters in the config file before running a job.

Load job
A transient (one-off) job that loads all requested topology data.

This job is started by the `taddm_observer_load_start.sh` script.

Procedure

To edit the parameters in the config file

1. Open the `$ASM_HOME/bin/taddm_observer_common.sh` configuration file and edit (at least) the following parameters:

   username
   TADDM user

   password
   TADDM password, in encrypted form. Use the `$ASM_HOME/bin/encrypt_password.sh` utility to generate an encrypted password.

   api_url
   TADDM API URL

   model_objects
   Optional

   List of supported TADDM model object names to be observed.

   Keep the default to let the observer fetch all the supported model objects. Supported model objects are `["AppServer", "ComputerSystem", "CPU", "StorageExtent", "L2Interface", "IpInterface", "IpAddress", "OperatingSystem", "Function", "SAPSystem", "Collection"]`

To start the Load job

2. To start the TADDM Observer load job, use the following command:
   ```
   $ASM_HOME/bin/taddm_observer_load_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need the TADDM topology data refreshed.

Results

You can now use data retrieved from the TADDM database to create topologies in the Agile Service Manager topology service.

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **`-h`** or **`--help`** to display help information, and with **`-v`** or **`--verbose`** to print out the

details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining VMware NSX Observer jobs

The VMware NSX Observer is installed as part of the core installation procedure. Use the VMware NSX Observer when you have VMware NSX installed in your environment to define jobs that dynamically load VMware NSX data for analysis by Netcool Agile Service Manager.

Before you begin

Important: The VMware NSX Observer supports VMware NSX versions 6.2 and 6.3.

Ensure you have the VMware NSX service details to hand, such as username, password, SSL TrustStore and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/vmwarensx-observer/swagger`

About this task

The VMware NSX Observer job extracts VMware NSX resource information via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`vmwarensx_observer_common.sh`
: The config file you use to customize VMware NSX Observer settings.

  The parameters defined here are then used by the `vmwarensx_observer_query_start.sh` script to trigger the VMware NSX Observer jobs.

  Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

You define and start the following job. You must edit the parameters in the config file before running this job.

Full Topology Upload job
: A transient (one-off) job that loads all requested topology data.

  This job is started by the `vmwarensx_observer_query_start.sh` script.

The VMware NSX Observer loads the following resources and their relationship into the Netcool Agile Service Manager core topology service:

- NSX Appliance
- vCenter Appliance
- NSX Controller
- Edge Router - Logical (Distributed) Router, Edge Service Gateway
- Virtual Machines
- Host
- VNIC

Procedure

To edit the parameters in the config file

1. Open the `vmwarensx_observer_common.sh` config file and edit (at least) the following parameters:

   `nsx_api_url`
   VMware NSX REST API endpoint

   `nsx_username`
   VMware NSX user name for REST API

   `nsx_password`
   VMware NSX user password for REST API

   Supply the VMware NSX user password in encrypted text.

   `nsx_tenant_name`
   VMware NSX tenant

   Set to 'default' if there is no specific tenant.

   `ssl_truststore_file`
   VMware NSX SSL trust store file for HTTPS authentication

   JKS is the supported format and the file is relative to the `$ASM_HOME/security` directory.

   `password_ssl_truststore_file`
   Password to decrypt an encrypted VMware NSX SSL trust store file

   Supply the VMware NSX SSL trust store password in encrypted format.

   Encryption requirement:

   The Load job requires passwords in encrypted form. To encrypt the nsx_password and password_ssl_truststore_file, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

   ```
   ./bin/encrypt_password.sh
   ```

   Enter and then confirm the password. The encryption utility will return an encrypted password.

To acquire VMware NSX SSL certificate and build SSL truststore

2. Use the following command to use OpenSSL to connect to VMware NSX over port 443, and extract a SSL Certificate from VMware NSX to a *<certificate_file_name>*`.crt` file.

   ```
   echo -n | openssl s_client -connect {VMware NSX IpAddress}:443 | sed -ne
   '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
   ```

3. Use the following Java keytool command to import the VMware NSX certificate file into a keystore and encrypt the keystore with a given password.

   ```
   keytool -import -v -trustcacerts -alias {VMware NSX Hostname}
   -file {certificate_file_name}.crt -keystore {keystore file name}
   -storepass {your password to encrypt keystore}
   ```

   Tip: You will need the following encryption information when editing `vmwarensx_observer_common.sh`

Table 45. Encryption parameters required for `vmwarensx_observer_common.sh`

keystore parameter	vmwarensx_observer_common.sh parameter
keystore password	password_ssl_truststore_file
keystore file name	ssl_truststore_file

4. Copy the keystore file ({`keystore file name`}) to the `$ASM_HOME/security` directory to complete the SSL setup.

To start the Load job

5. To start the VMware NSX Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/vmwarensx_observer_query_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need VMware NSX topology data refreshed.

What to do next

You can also use the following scripts:

vmwarensx_observer_query_stop.sh
Stops the Full Topology Upload job

vmwarensx_observer_job_list.sh
Lists the status of current jobs

vmwarensx_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **`-h`** or **`--help`** to display help information, and with **`-v`** or **`--verbose`** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining VMware vCenter Observer jobs

The VMware vCenter Observer is installed as part of the core installation procedure. Use the VMware vCenter Observer when you have VMware vCenter installed in your environment to define jobs that dynamically load VMware vCenter data for analysis by Netcool Agile Service Manager.

Before you begin

Important: The VMware vCenter Observer supports integration with VMware vCenter versions 6.5 and 6.7.

Ensure you have the VMware vCenter service details to hand, such as username, password, SSL TrustStore and URL.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/vmvcenter-observer/swagger`

About this task

The VMware vCenter Observer job extracts VMware vCenter resource information via REST. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

vmvcenter_observer_common.sh
The config file you use to customize VMware vCenter Observer settings.

The parameters defined here are then used by the `vmvcenter_observer_query_start.sh` script to trigger the VMware vCenter Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

You define and start the following job. You must edit the parameters in the config file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `vmvcenter_observer_query_start.sh` script.

The VMware vCenter Observer loads the following resources and their relationship into the Netcool Agile Service Manager core topology service:

- ESXi / ESX Hosts
- Virtual Machines
- VNICs
- Storage

Procedure

To edit the parameters in the config file

1. Open the `vmvcenter_observer_common.sh` config file and edit (at least) the following parameters:

   vcenter_api_url
   VMware vCenter REST API endpoint

   vcenter_username
   VMware vCenter user name for REST API

   vcenter_password
   VMware vCenter user password for REST API

   ssl_truststore_file
   VMware vCenter SSL trust store file for HTTPS authentication

   JKS is the supported format and the file is relative to `$ASM_HOME/data/vmcenter-observer`

   password_ssl_truststore_file
   Password to decrypt and encrypt VMware vCenter SSL trust store file

   Encryption requirement:

   The Load job requires passwords in encrypted form. To encrypt the vcenter_password and password_ssl_truststore_file, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

To acquire VMware vCenter SSL certificate and build SSL truststore

2. Use the following command to use OpenSSL to connect to VMware vCenter over port 443, and extract a SSL Certificate from VMware vCenter to a <*certificate_file_name*>.crt file.

   ```
   echo -n | openssl s_client -connect {VMware vCenter IpAddress}:443 | sed -ne
   '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
   ```

3. Use the following Java keytool command to import the VMware vCenter certificate file into a keystore and encrypt the keystore with a given password.

   ```
   keytool -import -v -trustcacerts -alias {VMware vCenter Hostname} -file
   {certificate_file_name}.crt -keystore {keystore file name}
   -storepass {your password to encrypt keystore}
   ```

   Tip: You will need the following encryption information when editing vmvcenter_observer_common.sh

Table 46. Encryption parameters required for vmvcenter_observer_common.sh

keystore parameter	vmvcenter_observer_common.sh parameter
keystore password	password_ssl_truststore_file
keystore file name	ssl_truststore_file

4. Copy the keystore file ({keystore file name}) to the $ASM_HOME/security directory to complete the SSL setup.

To start the Load job

5. To start the VMware vCenter Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/vmvcenter_observer_query_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need VMware vCenter topology data refreshed.

What to do next

You can also use the following scripts:

vmcenter_observer_query_stop.sh
Stops the Full Topology Upload job

vmcenter_observer_job_list.sh
Lists the status of current jobs

vmcenter_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Defining Zabbix Observer jobs

Using the Zabbix Observer functionality, you can load monitored servers and their associated network resources, and then visualize this data as a topology view in the Agile Service Manager UI. It is installed as part of the core installation procedure.

Before you begin

The Zabbix Observer supports Zabbix Version 4.0.3.

Ensure you have the Zabbix server details to hand, such as the username, password and SSL TrustStore.

Remember: Swagger documentation for the observer is available at the following default location: `https://<your host>/1.0/zabbix-observer/swagger`

About this task

A Zabbix Observer job extracts server information and its associated network resources from Zabbix via REST RPC. The Observer loads and updates the resources and their relationships within the Netcool Agile Service Manager core topology service.

`zabbix_observer_common.sh`
The configuration file you use to customize Zabbix Observer settings.

The parameters defined here are then used by the `zabbix_observer_load_start.sh` script to trigger the Zabbix Observer jobs.

Tip: Alternatively, you can set the appropriate environment variables. If an environment variable is set, it takes precedence over the config file settings.

You define and start the following job. You must edit the parameters in the config file before running this job.

Full Topology Upload job
A transient (one-off) job that loads all requested topology data.

This job is started by the `zabbix_observer_load_start.sh` script.

Procedure

To edit the parameters in the config file

1. Open the `zabbix_observer_common.sh` config file and edit (at least) the following parameters:

   `hostname`
   Zabbix hostname or ipaddreess

   `username`
   Zabbix user name

   `password`
   Zabbix user password

   Must be supplied in encrypted format

   `certificate`
   Optional certificate name. If provided, then a certificate file with the same name must exist in the `$ASM/security` directory.

ssl_truststore_file
Zabbix SSL trust store file for HTTPS authentication

JKS is the supported format and the file is relative to `$ASM_HOME/security`

truststore_password
Password to decrypt and encrypt Zabbix SSL trust store file

Must be encrypted

Encryption requirement:

The Load job requires passwords in encrypted form. To encrypt the password and truststore_password, run the `encrypt_password.sh` script in the `$ASM_HOME/bin` directory:

```
./bin/encrypt_password.sh
```

Enter and then confirm the password. The encryption utility will return an encrypted password.

connect_read_timeout_ms
Connection timeout in milliseconds (ms), for example '5000'.

To acquire Zabbix SSL certificate and build SSL truststore

2. Use the following command to use OpenSSL to connect to Zabbix, and extract an SSL Certificate from Zabbix to a *<certificate_file_name>*.crt file.

   ```
   echo -n | openssl s_client -connect {Zabbix IpAddress}:{SSL port | sed -ne
   '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ./{certificate_file_name}.crt
   ```

3. Use the following Java keytool command to import the Zabbix certificate file into a keystore and encrypt the keystore with a given password.

   ```
   keytool -import -v -trustcacerts -alias {Zabbix Hostname} -file
   {certificate_file_name}.crt -keystore {keystore file name}
   -storepass {your plain text password to encrypt keystore}
   ```

   Tip: You will need the following encryption information when editing `zabbix_observer_common.sh`

Table 47. Encryption parameters required for `zabbix_observer_common.sh`

keystore parameter	zabbix_observer_common.sh parameter
keystore password	truststore_password
keystore file name	ssl_truststore_file

4. Copy the keystore file ({keystore file name}) to the `$ASM_HOME/security` directory to complete the SSL setup.

To start the Load job

5. To start the Zabbix Observer Full Topology Upload job, use the following command:

   ```
   $ASM_HOME/bin/zabbix_observer_load_start.sh
   ```

   This job loads all requested topology data. Run this job whenever you need Zabbix topology data refreshed.

What to do next

You can also use the following scripts:

zabbix_observer_query_stop.sh
Stops the Full Topology Upload job

zabbix_observer_job_list.sh
Lists the status of current jobs

zabbix_observer_log_level.sh
Sets the log level

Remember: In addition to being configurable from the Observer Configuration UI, all on-prem observer jobs also have scripts to start and stop all available jobs, to list the status of a current job, and to set its logging levels. Scripts can be run with **-h** or **--help** to display help information, and with **-v** or **--verbose** to print out the details of the actions performed by the script, including the full cURL command. For the on-prem version of Agile Service Manager, observer scripts are configured for specific jobs by editing the script configuration files.

Chapter 5. Using Netcool Agile Service Manager

You use the Netcool Agile Service Manager UI to visualize your topology data. First you define a seed resource on which to base your view, then choose the levels of networked resources around the seed that you wish to display, before rendering the view. You can then further expand or analyze the displayed topology in real time, or compare it to previous versions within a historical time window.

The Netcool Agile Service Manager Topology Viewer has four toolbars and a visualization display area.

Navigation toolbar
: You use the navigation toolbar to select the seed resource, define the number of relationship hops to visualize from the seed resource, and specify the type of relationship hop traversal to make (either host-to-host, or element-to-element).

Resource filter toolbar
: You use the resource filter toolbar to apply entity- or relationship-type filters to the resources displayed in the topology.

Visualization toolbar
: You use the Visualization toolbar to customize the topology view, for example by zooming in and panning.

History toolbar
: You use the History toolbar to compare and contrast a current topology with historical versions.

Topology visualization panel
: You use the Topology visualization panel to view the topology, and access the resource nodes for further analysis performed via the context menu.

Logging into the UI (ICP)

You construct the Agile Service Manager ICP logon URL from the Netcool Operations Insight Helm release name.

Before you begin

Also see the following Netcool Operations Insight topic for more information: https://www.ibm.com/support/knowledgecenter/SSTPTP_1.5.0/com.ibm.netcool_ops.doc/soc/start/task/start_getting-started-icp.html

Tip: To prevent UI timeout errors on ICP, you can increase the timeout values for the topology, layout and search services. See the following troubleshooting topic for more details: “User interface timeout errors” on page 255

About this task

You login to the Agile Service Manager ICP installation using a URL of the following format (example):

`https://netcool.noi.icp-master.<your_host>/ibm/console`

Where *noi* is the Netcool Operations Insight Helm release name. Use the following command to retrieve the DASH URL:

```
helm status NOI helm release name --tls
```

Accessing the Topology Viewer in DASH (on-prem)

The Netcool Agile Service Manager UI consists of the topology viewer, which you access through DASH.

Before you begin

To access the topology viewer in DASH, you must have the appropriate user roles.

About this task

The Topology Viewer is accessed through an existing DASH deployment, giving you access to all its functionality. Typically, this would be part of an integration deployment that also gives you the use of other NOI applications.

Attention: During startup, topology services may try to connect to the Cassandra datastore before it has fully started, thereby causing an error, as also described in the related troubleshooting section. It will try again until the datastore is ready, and the error becomes void.

Procedure

1. Using a compatible browser, open DASH using the DASH URL. For example:
   `https://<DASH HOST>:<DASH PORT>/ibm/console/`
2. Login using your user credentials. DASH is displayed.
3. In DASH, open the **Incident** menu.
4. Click **Topology Viewer** (under the Agile Service Management heading). The Topology Viewer is displayed.

Results

The Netcool Agile Service Manager UI connects to the topology service, which provides the data needed to render the topology visualization. By default it refreshes the view dynamically every thirty seconds.

What to do next

Once you have accessed the Topology Viewer, you define the seed resource on which you want to base your topology view, and then choose the level of connected resources that you wish to render.

Accessing topologies via direct-launch URL string

You view specific topologies using a direct-launch URL by setting the topology navigation settings in the URL parameters to directly load a specific topology configuration within DASH. This gives you and others quick access to specific views. You can obtain a direct-launch URL string from a defined topology in Topology Viewer, or create it manually.

Before you begin

To obtain a direct-launch URL from the Topology Viewer accessed through DASH, you must have the appropriate DASH user roles. You must also complete the process of visualizing a topology. To share a direct-launch URL with others, they must be DASH users with the appropriate user roles.

Restriction: When launching a direct Netcool Agile Service Manager window in your browser, you **must** log onto DASH and keep the DASH window open as a separate tab in your browser. If you close the DASH browser tab, your DASH session will expire after the session timeout period, and you will be logged out of Netcool Agile Service Manager.

About this task

A typical URL starts from the base DASH URL, followed by more specific visualization parameters.

For a list of supported parameters, see the following topic: "Launch-in-context parameters" on page 222

Procedure

To obtain a topology URL from Topology Viewer in DASH

1. Once you have rendered a desired topology in the Topology Viewer, use the **Additional Actions** drop-down on the Navigation bar to obtain a direct launch URL string.

To define a URL manually

2. You can define a URL by editing the parameters, as shown in the following examples.

   Empty topology URL

   To open the Topology Viewer page directly with no configuration, use this type of URL.

   ```
   https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp
   ```

   Specify seed resource URL

   To open the page and create a visualization from a specific seed resource, use this format.

   This example sets the hop number as one, and the hop type as element-to-element. (Here you do not have to specify the hop type, as element-to-element is the default).

   ```
   https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceName=<name>
   ```

   Specify seed resource and hops URL

   To open the page and create a visualization from a specific seed resource with a specific number of hops and hop type, use this format.

   The **hopType** parameter is only required if you wish to use the host-to-host hop type (element-to-element is the default).

   ```
   https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceName=<name>
   &hops=<hops>&hopType=host
   ```

   Specify seed resource, hops and get neighbor expansions URL

   To open the page and create a visualization from a specific seed resource with a specific number of hops, as well as resource neighbors, use this format.

This example sets the hop type as element-to-element, and it uses the parameter **neighbourRequests**, which expects an array of space separated id strings.

```
https://<DASH HOST>:<DASH HOST>/ibm/console/URL>/inasm/topology.jsp?resourceName=
<name>&hops=<hops> &neighbourRequests=["<resource_id>" "<resource id>" ...]
```

Load topology using unique ID to set seed

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceUniqueId=
<unique id>
```

Load topology using advanced resource filters

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceFilter=
["Type1" "Type2"]
```

Load topology using advanced relationship filters

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?relationFilter=
["Type1" "Type2"]
```

Show topology at a given time point

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceName=<name>..
<any other configuration>..&time=<unixTimeMilliseconds>
```

Show topology in delta history mode, time for reference point and deltaTime is the point to delta against

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?resourceName=<name>..
<any other configuration>..&time=<unixTimeMilliseconds>&deltaTime=<unixTimeMilliseconds>
```

Load topology with side toolbar hidden

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?hideToolbar=true
```

Load topology with top search bar hidden

```
https://<DASH HOST>:<DASH PORT>/ibm/console/inasm/topology.jsp?hideSearch=true
```

Results

Having created direct-launch URL topology visualizations, you can save them for quick access to specific views, or share them with others to provide instant information.

Rendering (visualizing) a topology

You define the scope of the topology that you want to render by specifying a seed resource, the number of relationship hops surrounding that resource, as well as the types of hops. The topology service then supplies the data required to visualize the topology.

Before you begin

To visualize a topology, your topology service must be running, and your Observer jobs must be active.

About this task

You use this task to render a topology based on a specified seed resource.

Note: The UI has a default timeout set at 30 seconds. If service requests are not received in that time, a timeout message is shown, as in the following example:
`A time-out has occurred. No response was received from the Proxy Service within 30 seconds.`
See Topology render timeout for more information on addressing this issue.

Procedure

1. Access the topology viewer.

From the Navigation toolbar, perform the following actions:

2. Enter a search value for the resource you wish to use as seed in the **Resource Search** field. You can enter any value that may be indexed in the search service, such as the resource name or server. As you type in a search term, a drop-down list is displayed with suggested search terms that exist in the topology service.
   - If you select one of these suggested results, the Search Results page is displayed listing possible resource results.
   - You can expand a result in order to query the resource further and display more information. For each result, the name, status and other properties stored in the Elasticsearch engine are displayed.
   - Click the **Explore Topology** button next to a result to render the topology without setting any further display parameters.

   Tip: If the resource that you wish to find is unique and you are confident that it is the first result in the list of search results, then instead of selecting a result from the suggested search terms, you can choose to click the hairpin shortcut in the **Suggest** drop-down, which will render and display the topology for the closest matching resource.
3. Select a number between one and four to define the number of relationship hops to be visualized.
4. Choose one of the following hop types:
   - The **Element to Element** hop type performs the traversal using all element types in the graph.
   - The **Host to Host** hop type uses an aggregate traversal across elements with the entity type 'host'.
   - The **Element to Host** hop type provides an aggregated hop view like the 'Host to Host' type, but also includes the elements that are used to connect the hosts.
5. To prevent a large topology from being loaded, which can use considerable computational resources, you can set filters before rendering a topology.

   Open the Filter toolbar using the **Filter** toggle in the Topology Visualization toolbar (on the left), and apply the filters required. For more information on using filters, see the Filter the topology section in the 'Viewing a topology' topic.
6. Click **Render** to render the topology.

Results

The Netcool Agile Service Manager topology viewer connects to the topology service and renders the topology. By default the view is refreshed every thirty seconds, unless specified otherwise (by an administrator user).

Note: Topology render timeout: If you receive a timeout message, this may be due to a number of reasons:

- Large amounts of data being retrieved for complex topologies
- Too many hop counts specified
- Issues with the back-end services

Workarounds

- Check that all services are running smoothly. You can verify that the docker containers are running using the following command:

  ```
  $ASM_HOME/bin/docker-compose ps
  ```

  The system should return text indicating that all containers have a state of `Up`.
- Lower the hop count to reduce the service load. See the "Defining global settings" on page 204 topic for more information on customizing the maximum hop count.
- **An administrator user** can increase the default 30 seconds timeout limit by changing the following setting in the application.yml file:

  ```
  proxyServiceTimeout: 30
  ```

  You must restart DASH for the new timeout value to take effect:
  - To stop the DASH server, run `<DASH_PROFILE>/bin/stopServer.sh server1`
  - Once stopped, start the DASH server: `<DASH_PROFILE>/bin/startServer.sh server1`

What to do next

Next, you can refine and manipulate the view for further analysis.

Viewing a topology

Once you have rendered a topology, you can refine and manipulate the view.

Before you begin

To refine a topology, you must have previously defined a topology, as described in the "Rendering (visualizing) a topology" on page 166 topic.

Note: You can change a topology if and as required while viewing or refining an existing topology.

About this task

You can perform the following actions once you have rendered a topology:

View the topology
You can zoom in and out of the specific areas of the topology, and pan across it in various ways.

You can also auto-fit the topology into the available display window, draw a mini map, or redraw the entire topology.

Use the Update Manager
With auto-updates turned off, you can work with your current topology until you are ready to integrate the new resources into the view.

Filter resources
You can filter the types of resources displayed, or the types of relationships rendered.

Procedure

View a topology (created earlier)

1. From the Visualization toolbar below the Navigation toolbar, you can manipulate the topology using a number of visualization tools.

Select tool submenu
When you hover over the Select tool icon, a submenu is displayed from which you can choose the **Select**, **Pan** or **Zoom Select** tool.

Select tool
Use this icon to select individual resources using a mouse click, or to select groups of resources by creating a selection area (using click-and-drag).

Pan tool
Use this icon to pan across the topology using click-and-drag on a blank area of the visualization panel.

Zoom Select tool
Use this icon to zoom in on an area of the topology using click-and-drag.

Zoom In
Use this icon to zoom in on the displayed topology.

Zoom Out
Use this icon to zoom out of the displayed topology.

Zoom Fit
Use this icon to fit the entire topology in the current view panel.

Overview Toggle
Use this icon to create the overview mini map in the bottom right corner.

The mini map provides an overview of the entire topology while you zoom in or out of the main topology. The mini map displays a red rectangle to represent the current topology view.

Layout
Use this icon to recalculate, and then render the topology layout again.

You can choose from a number of layout types and orientations.

Layout 1
A layout that simply displays all resources in a topology without applying a specific layout structure.

Layout 2
A circular layout that is useful when you want to arrange a number of entities by type in a circular pattern.

Layout 3
A grouped layout is useful when you have many linked entities, as it helps you visualize the entities to which a number of other entities are linked. This layout helps to identify groups of interconnected entities and the relationships between them.

Layout 4
A hierarchical layout that is useful for topologies that contain hierarchical structures, as it shows how key vertices relate to others with peers in the topology being aligned.

Layout 5
A peacock layout is useful when you have many interlinked vertices, which group the other linked vertices.

Layout 6

A planar rank layout is useful when you want to view how the topology relates to a given vertex in terms of its rank, and also how vertices are layered relative to one another.

Layout 7

A rank layout is useful when you want to see how a selected vertex and the vertices immediately related to it rank relative to the remainder of the topology (up to the specified amount of hops). The root selection is automatic.

For example, vertices with high degrees of connectivity outrank lower degrees of connectivity. This layout ranks the topology automatically around the specified seed vertex.

Layout 8

A root rank layout similar to layout 7, except that it treats the selected vertex as the root. This layout is useful when you want to treat a selected vertex as the root of the tree, with others being ranked below it.

Ranks the topology using the selected vertex as the root (root selection: Selection)

Layout orientation

For layouts 4, 6, 7 and 8, you can set the following layout orientations:

- Top to bottom
- Bottom to top
- Left to right
- Right to left

Filter Toolbar toggle

Use this icon to display or hide the filter toolbar. You can filter resources that are displayed in the topology, or set filters before rendering a topology.

If a filter has been applied to a displayed topology, the text 'Filtering applied' is displayed in the status bar at the bottom of the topology.

History toggle

Use this to open and close the Topology History toolbar. The topology is displayed in history mode by default.

Configure Refresh Rate

When you hover over the **Refresh Rate** icon, a submenu is displayed from which you can configure the auto-update refresh rate.

You can pause the topology data refresh, or specify the following values: 10 seconds, thirty seconds (default), one minute, or five minutes.

Resource display conventions

Deleted: A minus icon shows that a resource has been deleted since last rendered.

Displayed when a topology is updated, and in the history views.

Added: A purple plus (+) icon shows that a resource has been added since last rendered.

Displayed when a topology is updated, and in the history views.

Added (neighbors): A blue asterisk icon shows that a resource has been added using the 'get neighbors' function.

Use the Update Manager

2. If auto-updates have been turned off, the Update Manager informs you if new resources have been detected. It allows you to continue working with your current topology until you are ready to integrate the new resources into the view. The Update Manager is displayed in the bottom right of the screen.

   Show details
   Displays additional resource information.

   Render
   Integrates the new resources into the topology.

   Choosing this option will recalculate the topology layout based on your current display settings, and may therefore adjust the displayed topology significantly.

   Cogwheel icon
   When clicked, provides you with quick access to change your user preferences:

   - **Enable auto-refresh:** Switches auto-refresh back on, and disables the Update Manager.
   - **Remove deleted resources:** Removes the deleted resources from your topology view when the next topology update occurs.

   Hide Reduces the Update Manager to a small purple icon that does not obstruct your current topology view.

   When you are ready to deal with the new resources, click on the icon to display the Update Manager again.

Modify a topology

3. The displayed topology consists of resource nodes and the relationship links connecting the resources. You can interact with these nodes and links using the mouse functionality.

   Dragging a node
   Click and drag a node to move it.

   Selecting a node
   Selection of a node highlights the node, and emphasizes its first-order connections by fading all other resources.

   Context menu (right-click)
   You open the context menu using the right-click function. The context menu provides access to the resource-specific actions you can perform.

   For resource entities, you can perform the following:

   Resource Details
   When selected, displays a dialog that shows all the current stored properties for the specified resource in table format.

   Resource Status
   If statuses related to a specific resource are available, the resource will be marked with an icon depicting the status severity level, and the Resource Status option will appear in the resource context menu.

When selected, Resource Status displays a dialog that shows the time-stamped statuses related to the specified resource in table format. The Severity, Time, and State columns can be sorted, and the moment that Resource Status was selected is also time-stamped.

In addition, if any status tools have been defined, the status tool selector (three dots) is displayed next to the resource's statuses. Click the status tool selector to display a list of any status tools that have been defined, and then click the specific tool to run it. Status tools are only displayed for the states that were specified when the tools were defined.

The **state** of a status is either 'open', 'clear', or 'closed'.

The **severity** of a status ranges from 'clear' (white tick on a green square) to 'critical' (white cross on a red circle).

Table 48. Severity levels

Icon	Severity
	clear
	indeterminate
	information
	warning
	minor
	major
	critical

Comments

When selected, this displays any comments recorded against the resource.

By default, resource comments are displayed by date in ascending order. You can sort them in the following way:

- Oldest first
- Newest first
- User Id (A to Z)
- User Id (Z to A)

Users with the inasm_operator role can view comments, but not add any. Users with inasm_editor or inasm_admin roles can also add new comments. See the “Configuring DASH user roles” on page 21 topic for more information on assigning user roles.

To add a new comment, enter text into the New Comment field, and then click **Add Comment** to save.

Get Neighbors

When selected, opens a menu that displays the resource types

of all the neighboring resources. Each resource type lists the number of resources of that type, as well as the maximum severity associated with each type.

You can choose to get all neighbors of the selected resource, or only the neighbors of a specific type. This lets you expand the topology in controlled, incremental steps.

Selecting **Get Neighbors** overrides any existing filters.

You can **Undo** the last neighbor request made.

Follow Relationship
When selected, opens a menu that displays all adjacent relationship types.

Each relationship type lists the number of relationships of that type, as well as the maximum severity associated with each type.

You can choose to follow all relationships, or only the neighbors of a specific type.

Show last change in timeline
When selected, will display the history timeline depicting the most recent change made to the resource.

Show first change in timeline
When selected, will display the history timeline depicting the first change made to the resource.

Recenter View
When selected, this updates the displayed topology with the specified resource as seed.

Filter the topology

4. Open the Resource Filter toolbar using the **Filter** toggle in the Topology Visualization toolbar. From here, you can apply filters to the topology in order to refine the types of resources or relationships displayed. The Filter toolbar is displayed as a panel on the right-hand side of the page, and consists of a **Simple** and an **Advanced** tab. If selected, each tab provides you with access to lists of Resource types and Relationship types. Only types relevant to your topology are displayed, for example **host**, **ipaddress** or **operatingsystem**, although you can use the **Show all types** toggle to view all of them.

   Simple tab
   When you use the Simple tab to filter out resource or relationship types, all specified types are removed from view, including the seed resource.

   It **only** removes the resources matching that type, leaving the resources below, or further out from that type, based on topology traversals.

   By default, all types are **On**. Use the **Off** toggle to remove specific types from your view.

   Advanced tab
   The Advanced tab performs a server-side topology-based filter action.

   It removes the resources matching that type, **as well as** all resources below that type.

However, the seed resource is **not** removed from view, even if it is of a type selected for removal.

Tips

Reset or invert all filters: Click **Reset** to switch all types back on, or click **Invert** to invert your selection of types filtered.

Hover to highlight: When a topology is displayed, hover over one of the filtering type options to highlight them in the topology.

Viewing topology history

You can view a topology dynamically, or use the history timeline function to compare and contrast the current topology with historical versions.

Before you begin

To refine a topology, you must have previously defined a topology, as described in the "Rendering (visualizing) a topology" on page 166 topic.

Note: You can change a topology if and as required while viewing or refining an existing topology.

About this task

Tip: The topology is displayed in history mode by default.

Procedure

1. Open the Topology History toolbar by clicking the **History** toggle in the Topology Visualization toolbar (on the left).
2. You can display and refine topology history in a number of ways.

   Update mode

   The topology is displayed in update mode by default with Delta mode set to **Off**.

   While viewing the timeline in update mode with Delta mode set to **On**, any changes to the topology history are displayed on the right hand side of the timeline, with the time pins moving apart at set intervals. By clicking **Render**, you reset the endpoint to 'now' and the pins form a single line again.

   While viewing the timeline in update mode with Delta mode set to **Off**, only a single pin is displayed.

   Delta mode

   You toggle between delta mode **On** and **Off** using the Delta switch above the topology.

   When Delta mode is **On** with Update mode also **On**, differences in topology are displayed via purple plus or minus symbols next to the affected resource.

   When Delta mode is **On** with History mode **On** (that is, Update mode set to **Off**), you can compare two time points to view differences in topology.

Lock time pin

Click the **Lock** icon on a time pin's head to lock a time point in place as a reference point, and then use the second time slider to view topology changes.

History timeline

You open the Topology History toolbar using the **History** toggle in the Topology Visualization toolbar (on the left).

You use the time pins to control the topology shown. When you move the pins, the topology updates to show the topology representation at that time.

While in delta mode you can move both pins to show a comparison between the earliest pin and the latest. The timeline shows the historic changes for a single selected resource, which is indicated in the timeline title. You can lock one of the time pins in place to be a reference point.

When you first display the history timeline, coach marks (or tooltips) are displayed, which contain helpful information about the timeline functionality. You can scroll through these, or switch them off (or on again) as required.

To view the timeline for a different resource, you click on it, and the heading above the timeline changes to display the name of the selected resource. If you click on the heading, the topology centers (and zooms into) the selected resource.

The history timeline is displayed above a secondary time bar, which displays a larger time segment and indicates how much of it is depicted in the main timeline. You can use the jump buttons to move back and forth along the timeline, or jump to the current time.

You can use the time picker, which opens a calendar and clock, to move to a specific second in time.

To view changes made during a specific time period, use the two time sliders to set the time period. You can zoom in and out to increase or decrease the granularity using the + and - buttons on the right, or by double-clicking within a time frame. The most granular level you can display is an interval of one second. The granularity is depicted with time indicators and parallel bars, which form 'buckets' that contain the recorded resource change event details.

The timeline displays changes to a resource's state, properties, and its relationships with other resources. These changes are displayed through color-coded bars and dash lines, and are elaborated on in a tooltip displayed when you hover over the change. You can exclude one or more of these from display.

Resource state changes

The timeline displays the number of state changes a resource has undergone.

Resource property changes

The timeline displays the number of times that resource properties were changed.

Each time that property changes were made is displayed as one property change event regardless of whether one or more properties were changed at the time.

Resource relationship changes

The number of relationships with neighboring resources are displayed, and whether these were changed.

The timeline displays when relationships with other resources were changed, and also whether these changes were the removal or addition of a relationship, or the modification of an existing relationship.

Rebuilding a topology

Once you have rendered a topology, you can search for (or define) a new seed resource and build a topology around it, change the number of hops rendered, and switch between element-to-element, host-to-host and element-to-host hop types.

Before you begin

To refine a topology, you must have previously defined a topology, as described in the "Rendering (visualizing) a topology" on page 166 topic.

Note: You can change a topology if and as required while viewing or refining an existing topology.

About this task

Tip: For information on re-indexing the Search service, see the 'Re-indexing Search' information in the task troubleshooting section of this topic.

Procedure

Change the hops, search for a resource, or rebuild a topology

From the Navigation toolbar, you can again search for a resource around which to build a topology, change the number of hops and the type of hop, and re-render the topology.

Resource Search

The seed resource of the topology visualization.

You define the seed resource around which a topology view is rendered by searching for a seed in the resource search. As you type in a search term related to the resource that you wish to find, such as name or server, a drop-down list is displayed with suggested search terms that exist in the topology service.

If you select one of the suggested results, the Search Results page is displayed listing possible resource results. For each result, the name, type and other properties stored in the Elasticsearch engine are displayed.

You can expand a result in order to query the resource further and display more detailed, time-stamped information, such as its state and any associated severity levels, or when the resource was previously updated or replaced (or deleted).

You can click the **Explore Topology** button next to a result to render the topology.

If the resource that you wish to find is unique and you are confident that it is the first result in the list of search results, then instead of selecting a

result from the suggested search terms, you can choose to click the shortcut in the **Suggest** drop-down, which will render and display the topology for the closest matching resource.

Topology Search
If you conduct a Resource Search with a topology already loaded, the search functionality searches the loaded topology as well as the topology database. As you type in a search term, a drop-down list is displayed that includes suggested search results from the displayed topology listed under the **In current view** heading.

If you hover over a search result in this section, the resource is highlighted in the topology window.

If you click on a search result, the topology view zooms in on that resource and closes the search.

No. Hops
The number of relationship hops to visualize from the seed resource, with the default set at 'one'.

You define the number of relationship hops to be performed, which can be from one to four, unless this setting has been customized. See the "Defining global settings" on page 204 topic for more information on customizing the maximum hop count.

Type of Hop
The type of graph traversal used.

The two options are:

Element to Element hop type
This type performs the traversal using all element types in the graph.

Host to Host hop type
This type generates a view showing host to host connections.

Element to Host hop type
This type provides an aggregated hop view like the Host to Host type, but also includes the elements that are used to connect the hosts.

Tip: The URL captures the hopType as 'e2h'. When launching a view using a direct URL, you can use the `hopType=e2h` URL parameter.

Render
This performs the topology visualization action, rendering the topology based on the settings in the navigation toolbar.

Preemptive filtering: To prevent a large topology from being loaded, which can use considerable computational resources, you can set filters before rendering a topology.

Once rendered, the topology will refresh on a 30 second interval by default. You can pause the auto-update refresh, or select a custom interval.

Tip: The UI can time out if a large amount of data is being received. See the timeout troubleshooting section in the following topic for information on how to address this issue, if a timeout message is displayed: "Rendering (visualizing) a topology" on page 166

Performing topology administration

From the Topology Viewer, you can obtain direct-launch URLs, perform a system health check, and set user preferences.

Before you begin

Access the Topology Viewer.

About this task

You can perform the following admin actions:

Save direct launch URL
You can copy and save a URL to quickly access a currently defined topology view.

View system health
You can view your system's health.

Set user preferences
You can set user preferences that define the default settings for rendering your topology.

Procedure

You perform the following actions from the **Navigation bar** > **Additional actions** menu.

Additional actions > Obtain Direct URL
Open the **Additional actions** (**...**) drop-down menu, and then use the **Obtain Direct URL** option to display the Direct Topology URL dialog.

The displayed URL captures the current topology configuration, including layout type (layout orientation is not tracked).

Click **Copy** to obtain a direct-launch URL string, then click **Close** to return to the previous screen.

Use the direct-launch URL for quick access to a given topology view within DASH.

Tip: You can share this URL with all DASH users with the required permissions.

Additional actions > View System Health
Open the **Additional actions** (**...**) drop-down menu, and then use the **View System Health** option to access your Netcool Agile Service Manager deployment's system health information.

Additional actions > Edit User Preferences
Open the **Additional actions** (**...**) drop-down menu, and then use the **Edit User Preferences** option to access the User Preferences window. Click **Save**, then **Close** when done.

You can customize the following user preferences to suit your requirements:

Updates

Default auto refresh rate (seconds)
The rate at which the topology will be updated.

The default value is 30.

You must reopen the page before any changes to this user preference take effect.

Maximum number of resources to load with auto refresh enabled

When the resource limit set here is reached, auto-refresh is turned off.

The maximum value is 2000, which is also set as the default.

Tip: If you find that the default value is too high and negatively impacts your topology viewer's performance, reduce this value.

Auto render new resources

Enable this option to display new resources at the next scheduled or ad-hoc refresh as soon as they are detected.

Remove deleted topology resources

Enable this option to remove deleted resources at the next scheduled or ad-hoc refresh.

Layout

Set **Default layout type** including the layout orientation for some of the layout types. You can also configure a default layout in User Preferences.

You can choose from a number of layout types, and also set the orientation for layouts 4, 6, 7 and 8.

Tip: A change to a layout type is tracked in the URL (layout orientation is not tracked). You can manually edit your URL to change the layout type display settings.
The following numbered layout types are available:

Layout 1

A layout that simply displays all resources in a topology without applying a specific layout structure.

Layout 2

A circular layout that is useful when you want to arrange a number of entities by type in a circular pattern.

Layout 3

A grouped layout is useful when you have many linked entities, as it helps you visualize the entities to which a number of other entities are linked. This layout helps to identify groups of interconnected entities and the relationships between them.

Layout 4

A hierarchical layout that is useful for topologies that contain hierarchical structures, as it shows how key vertices relate to others with peers in the topology being aligned.

Layout 5
A force-directed (or 'peacock') layout is useful when you have many interlinked vertices, which group the other linked vertices.

Layout 6
A planar rank layout is useful when you want to view how the topology relates to a given vertex in terms of its rank, and also how vertices are layered relative to one another.

Layout 7
A rank layout is useful when you want to see how a selected vertex and the vertices immediately related to it rank relative to the remainder of the topology (up to the specified amount of hops). The root selection is automatic.

For example, vertices with high degrees of connectivity outrank lower degrees of connectivity. This layout ranks the topology automatically around the specified seed vertex.

Layout 8
A root rank layout similar to layout 7, except that it treats the selected vertex as the root. This layout is useful when you want to treat a selected vertex as the root of the tree, with others being ranked below it.

Ranks the topology using the selected vertex as the root (root selection: Selection)

Layout orientation
For layouts 4, 6, 7 and 8, you can set the following layout orientations:

- Top to bottom
- Bottom to top
- Left to right
- Right to left

Misc

Information message auto hide timeout (seconds)
The number of seconds that information messages are shown for in the UI.

The default value is 3.

Tip: If you are using a screen reader, it may be helpful to increase this value to ensure that you do not miss the message.

Screen reader support for graphical topology
You can enable the display of additional Help text on screen elements, which can improve the usability of screen readers.

You must reopen the page before any changes to this user preference take effect.

Enhanced client side logging, for problem diagnosis
If enabled, additional debug output is generated, which you can use for defect isolation.

Tip: Use this for specific defect hunting tasks, and then disable it again. If left enabled, it can reduce the topology viewer's performance.

You must reopen the page before any changes to this user preference take effect.

Chapter 6. Administration

Use the following topics to understand administration tasks, such as monitoring system health and logging.

Configuring core services authentication

To customize secure access to the core services, you can change the default authentication method the UI uses when connecting to core services from basic authentication to LDAP. You can also encrypt the password and generate a new password encryption key.

Configuring the authentication method for the UI when connecting to core services

You can change the authentication method used by the UI to access core services.

Before you begin

Agile Service Manager must be installed and running.

To configure the authentication method for the UI when connecting to core services, you use the username and password supplied during the Agile Service Manger UI installation (after the core installation), or use the default `asm` username and password.

About this task

Agile Service Manger services have access to two mechanisms of authentication:

Basic authentication (default)
: Environment variable based using the **ASM_USER** and **ASM_PASS** variables configured in `$ASM_HOME/.env`

LDAP
: LDAP authentication configured using environment variables in `$ASM_HOME/.env`

Procedure

Switch authentication method

1. The values of the **ASM_AUTHENTICATOR** parameter in `.env` determine the authentication method:
   - `com.ibm.itsm.topology.service.auth.BasicAuthenticator`
   - `com.ibm.itsm.topology.service.auth.LdapAuthenticator`

Configure authentication access details

2. Once you have defined the authentication method, configure access details:
   - If using **basic authentication**, edit `$ASM_HOME/.env` and change the values for **ASM_USER** and **ASM_PASS**
   - If using **LDAP authentication**, edit `$ASM_HOME/.env` and change the following values:

Service name
LDAP_SERVICE_NAME: ${LDAP_SERVICE_NAME:-localhost}

Service port
LDAP_SERVICE_PORT: ${LDAP_SERVICE_PORT:-389}

SSL LDAP_USE_SSL: ${LDAP_USE_SSL:-0}

LDAP base
LDAP_BASE_DN: ${LDAP_BASE_DN:-dc=example,dc=com}

Supply authentication credentials to Swagger

3. To run REST requests via Swagger, you must use the configured username and password.

   Tip: When changing user credentials in Swagger, the interface sometimes prevents you from logging out.
   Workaround: Restart your browser.

Update application.yml on the UI server

Note: You only need to update the application.yml file as described in the following steps if you have added or changed user credentials used by the UI.

4. Open the application.yml file in a text editor and change the `proxyServicePassword` property value to the encrypted version of the password.
5. Change the `passwordEncryption` property to `true`

   Important: You must set the passwordEncryption property in the application.yml file to true if the core services password has been encrypted, or the UI server will not be able to authenticate and will therefore be unable to access any topology data.
6. Restart DASH to allow the changes to take effect.

   Stop the DASH server
   `<DASH_PROFILE>/bin/stopServer.sh server1`

   Restart the DASH server
   `<DASH_PROFILE>/bin/startServer.sh server1`

Related information:

https://httpd.apache.org/docs/2.4/programs/htpasswd.html

Encrypting the password for UI access to core services

You can encrypt the password used for connecting to the Agile Service Manager services using the `encrypt_password` tool located in the `INASM_UI_HOME/bin` directory.

Before you begin

This password used by the Agile Service Manager UI to connect to the core services is stored as plain text in the `INASM_UI_HOME/etc/application.yml` configuration file with the other connection parameters. To improve security, you can encrypt this password.

Ensure you have the core services password to hand when completing this procedure.

Tip: Encryption uses a default 128-bit FIPS-compliant encryption key, which is supplied with the Agile Service Manager UI during installation. You can generate a

new encryption key to replace the existing key, which is described in the following topic: “Generating a new password encryption key” on page 186

About this task

To store the password in the application.yml file in an encrypted form, you first run an encryption tool to obtain an encrypted form of the password, and then update the application.yml file with the encrypted text.

File location examples

Application configuration file location:
`/opt/IBM/netcool/gui/inasm/etc/application.yml`

Encryption tool location:
`/opt/IBM/netcool/gui/inasm/bin`

Procedure

To encrypt the password

1. From a command console log into the DASH server (which also hosts the ASM UI), and navigate to the Agile Service Manager scripts directory. The default location of the scripts directory is `INASM_UI_HOME/bin`
2. Run the encrypt_password tool, using the following platform-specific commands:

Option	Description
Windows	`encrypt_password`
Unix	`./encrypt_password.sh`

3. When prompted, enter the password, and then retype it when prompted again. The encryption tool will display an encrypted version of the password.
4. Copy the encrypted password.

To update the application.yml file with the encrypted password

5. Open the application.yml file in a text editor and change the `proxyServicePassword` property value to the encrypted version of the password.
6. Change the `passwordEncryption` property to `true`

   Important: You must set the passwordEncryption property in the application.yml file to true if the core services password has been encrypted, or the UI server will not be able to authenticate, and will therefore be unable to access any topology data.
7. Restart DASH to allow the changes to take effect.

   Stop the DASH server

   `<DASH_PROFILE>/bin/stopServer.sh server1`

   Restart the DASH server

   `<DASH_PROFILE>/bin/startServer.sh server1`

Results

The Agile Service Manager core services password is now being stored in the application.yml file in a more secure, encrypted format.

Related tasks:

"Generating a new password encryption key"
You can generate a new password encryption key using the generate_key tool located in the INASM_UI_HOME/bin directory. You can then use that new key file to encrypt passwords.

Generating a new password encryption key

You can generate a new password encryption key using the generate_key tool located in the INASM_UI_HOME/bin directory. You can then use that new key file to encrypt passwords.

Before you begin

You only generate a new password encryption key if you are encrypting the core services password, but do not want to use the default encryption key.

About this task

You can encrypt the core services password that is stored in the application.yml configuration file, as described in the following topic: "Encrypting the password for UI access to core services" on page 184

Encryption uses a default 128-bit FIPS-compliant encryption key, which is supplied with the Agile Service Manager UI during installation, and is located here:
INASM_UI_HOME/security/crypto.key

You can use a key generation tool to generate a new encryption key to replace the existing key. You can then re-encrypt the core services password and update the application.yml file.

File location examples

Encryption key location:
/opt/IBM/netcool/gui/inasm/security/crypto.key

Encryption key generation tool location:
/opt/IBM/netcool/gui/inasm/bin

Application configuration file location:
/opt/IBM/netcool/gui/inasm/etc/application.yml

Procedure

To generate a new encryption key

1. From a command console log into the DASH server (which also hosts the ASM UI), and navigate to the Agile Service Manager scripts directory. The default location of the scripts directory is INASM_UI_HOME/bin
2. Run the generate_key tool, using the following platform-specific commands:

Option	Description
Windows	generate_key
Unix	./generate_key.sh

   A system message is displayed to warn you that the existing key file will be overwritten if you proceed.
3. At the prompt, type y to continue. The key generation tool generates a new key and overwrites the existing key.

CAUTION:
If you generate a new encryption key, but then do not use it to re-encrypt your password, the existing password in the application.yml file will be incompatible with the new key. The next time the DASH server is restarted, the UI server will not be able to authenticate with the core services, and will therefore be unable to access any topology data.

To re-encrypt the password

4. Still from the command console in the Agile Service Manager scripts directory, run the encrypt_password tool:

Option	Description
Windows	`encrypt_password`
Unix	`./encrypt_password.sh`

   The encryption tool will display an encrypted version of the password.
5. Copy the encrypted password.

To update the application.yml file with the encrypted password

6. Open the application.yml file in a text editor and change the `proxyServicePassword` property value to the new encrypted password.
7. Ensure that the `passwordEncryption` property is still set to `true`
8. Restart DASH to allow the changes to take effect.

   Stop the DASH server

   `<DASH_PROFILE>/bin/stopServer.sh server1`

   Restart the DASH server

   `<DASH_PROFILE>/bin/startServer.sh server1`

Results

The Agile Service Manager core services password stored in the application.yml file has been updated with a new encrypted version.

Important: Do not move or rename the crypto.key file, or the UI application will be unable to find it in order to process the encrypted password, and so will be unable to authenticate with the Agile Service Manager core services.

Related tasks:

"Encrypting the password for UI access to core services" on page 184
You can encrypt the password used for connecting to the Agile Service Manager services using the `encrypt_password` tool located in the `INASM_UI_HOME/bin` directory.

Configuring SSL between the UI and the proxy service

The Agile Service Manager UI communicates with the proxy service via HTTPS (TLS). The UI uses a default SSL trust store containing the proxy service certificate to encrypt the communications between them.

The default location for the trust store file is: `$NCHOME/inasm/security/truststore.p12`

You can perform the following administrative tasks:

- Change the password for the Agile Service Manager UI trust store.
- Update the signer certificate in the Agile Service Manager UI trust store.

- Use the default WebSphere 'NodeDefaultTrustStore' trust store instead of the one provided by Agile Service Manager (truststore.p12).
- Use a custom trust store file instead of the one provided (truststore.p12).

Changing the password for the Agile Service Manager UI trust store

The default SSL trust store 'truststore.p12' installed with the Agile Service Manager UI has a password of 'asmtrust', which you can change as described here.

Procedure

1. Locate the Java JRE being used by WebSphere (either Java 7 or Java 8). For a typical DASH installation, for example, the location of the Java 7 JRE is `/opt/IBM/WebSphere/AppServer/java_1.7_64/jre`
2. From the JRE bin subdirectory, change the password of the ASM UI trust store using one of the following utilities:
   - ikeyman (GUI)
   - keytool (CLI)

   a. Use `$NCHOME/inasm/security/truststore.p12` as the path of the trust store file.

      Note: Replace `$NCHOME` with the actual path, for example: `/opt/IBM/netcool/gui`

   b. Use `asmtrust` as the initial trust store password.

   c. Use `PKCS12` as the trust store type.
3. Update the **`sslTrustStorePassword`** setting in the `application.yml` file to store the updated password. See the following topic for more information: "Editing the application settings file" on page 22
4. Restart DASH to allow the changes to take effect.

Changing the certificate for the Agile Service Manager UI trust store

You can replace the Agile Service Manager signer certificate used by Nginx with your own certificate. To make HTTPS connections to the proxy service, you then also update the Agile Service Manager UI trust store to contain the new Nginx certificate.

Before you begin

For the on-prem version of Agile Service Manager, you must configure Nginx to use a custom signed certificate first. To use HTTPS, Nginx requires a private key file and a certificate file to be present.

1. Edit the 'server' section of the `/opt/ibm/netcool/asm/etc/nginx/nginx.conf` file as follows:

```
server {
   listen   8443 ssl;
   server_name   localhost;
      ssl_certificate /opt/ibm/netcool/asm/security/asm-nginx.crt;
      ssl_certificate_key /opt/ibm/netcool/asm/security/asm-nginx.key;
   ssl_protocols TLSv1.1 TLSv1.2;
...
```

Nginx will expect to find a private key file (asm-nginx.key) and a certificate file (asm-nginx.crt) in the `/opt/ibm/netcool/asm/security` directory. The CN (Common Name) in the certificate should be the **fqdn** of the host machine where Agile Service Manager is running. The asm-nginx.key and asm-nginx.crt files will be generated automatically after the installation of Agile Service Manager.

2. Restart the proxy service when done:
   ```
   $ASM_HOME/bin/docker-compose restart proxy
   ```

Note: If you want to replace the generated files with your own certificate and key files, copy them to the `/opt/ibm/netcool/asm/security/` directory, and then edit the 'ssl_certificate' and 'ssl_certificate_key' properties of the configuration file accordingly. Ensure the CN is the **fqdn** of the host machine.

About this task

If you replace the default Agile Service Manager UI certificate used by Nginx with another, you must update the UI trust store to contain the new Nginx certificate, so that the UI can continue to connect to the proxy service.

Procedure

1. Obtain the new Nginx certificate. You can obtain the certificate by one of the following methods:
   - By file transfer from the server hosting the Nginx certificate.
   - By web browser from Nginx, by viewing and then exporting the certificate to file.
     Use the following URL: `https://`*`proxy-service-host`*`:`*`proxy-service-port`*
     For example: `https://asm-host:443`
2. Locate the Java JRE being used by WebSphere (either Java 7 or Java 8). For example:
   `/opt/IBM/WebSphere/AppServer/java_1.7_64/jre`
3. From the JRE bin subdirectory, import the new Nginx certificate into the Agile Service Manager UI trust store using one of the following utilities:
   - ikeyman (GUI)
   - keytool (CLI)

   a. Use `$NCHOME/inasm/security/truststore.p12` as the path of the trust store file.

      Note: Replace `$NCHOME` with the actual path, for example: `/opt/IBM/netcool/gui`

   b. Use `asmtrust` as the initial trust store password.

   c. Use `PKCS12` as the trust store type.
4. Optional: Remove the previous certificate using either the ikeyman or keytool utility.

   Tip: The default, now obsolete certificate has the alias **asm-ca**.
5. Restart DASH to allow the changes to take effect.

Changing the default trust store to the WebSphere trust store

Instead of using the default SSL trust store (truststore.p12) installed with the Agile Service Manager UI, you can use the default WebSphere trust store.

Procedure

1. Locate the Java JRE being used by WebSphere (either Java 7 or Java 8). For example:
   `/opt/IBM/WebSphere/AppServer/java_1.7_64/jre`
2. From the JRE bin subdirectory, export the Agile Service Manager certificate from the Agile Service Manager trust store to a file using one of the following utilities:
   - ikeyman (GUI)
   - keytool (CLI)

   a. Use `$NCHOME/inasm/security/truststore.p12` as the path of the trust store file.

      Note: Replace `$NCHOME` with the actual path, for example:
      `/opt/IBM/netcool/gui`

   b. Use `asmtrust` as the initial trust store password.

   c. Use `PKCS12` as the trust store type.

   d. Use `asm-ca` as the certificate alias, if the original Nginx certificate is being used. If you have changed the trust store certificate, use its alias instead.
3. As the admin user, log into DASH and open the WebSphere Administration Console: **Console Settings** > **WebSphere Administrative Console** > **Launch WebSphere Administrative Console**
4. In the WebSphere Administrative Console, navigate to the certificates: **Security** > **SSL certificate and key management** > **Key stores and certificates** > **NodeDefaultTrustStore** > **Signer certificates**
5. Click **Add** to import the ASM certificate into the NodeDefaultTrustStore trust store from the certificate file saved earlier.
6. When prompted, save your changes to the WebSphere master configuration.

   Important: The Agile Service Manager proxy service supports only TLS v1.1 and TLS v1.2 protocols for HTTPS communications. If your default WebSphere SSL settings are set to use a different version of SSL or TLS, update the WebSphere **Quality of Protection** settings to specify the use of TLS 1.1 or 1.2. See the WebSphere documentation for information on how to change the default SSL configuration.
7. Remove any values that are already specified for the following parameters in the Agile Service Manager UI application.yml file:
   - sslTrustStorePath
   - sslTrustStorePassword
   - sslTrustStoreType

   When left blank, these parameters direct Agile Service Manager to use the default WebSphere trust store. See the following topic for more information: "Editing the application settings file" on page 22
8. Restart DASH to allow the changes to take effect.

Changing the default trust store to a custom trust store

Instead of using the default SSL trust store (truststore.p12) installed with the Agile Service Manager UI, you can use a custom trust store.

Procedure

1. Locate the Java JRE being used by WebSphere (either Java 7 or Java 8). For example:
   `/opt/IBM/WebSphere/AppServer/java_1.7_64/jre`
2. From the JRE bin subdirectory, export the Agile Service Manager certificate from the Agile Service Manager trust store to a file using one of the following utilities:
   - ikeyman (GUI)
   - keytool (CLI)

   a. Use `$NCHOME/inasm/security/truststore.p12` as the path of the trust store file.

      Note: Replace `$NCHOME` with the actual path, for example:
      `/opt/IBM/netcool/gui`

   b. Use `asmtrust` as the initial trust store password.

   c. Use `PKCS12` as the trust store type.

   d. Use `asm-ca` as the certificate alias, if the original Nginx certificate is being used. If you have changed the trust store certificate, use its alias instead.
3. Use either the ikeyman or keytool utility to create a new keystore to act as the new custom Agile Service Manager trust store. You **must** use either PKCS12 or JKS key store types (other types are not supported).
4. Use either the ikeyman or keytool utility to import the Agile Service Manager certificate from the certificate file saved earlier into the new trust store.
5. Update the following parameters in the Agile Service Manager UI application.yml file to match the values for the newly-created trust store:
   - sslTrustStorePath
   - sslTrustStorePassword
   - sslTrustStoreType

   See the following topic for more information: “Editing the application settings file” on page 22
6. Required: Ensure that the new trust store file has sufficient file permissions to be readable by the user that starts the DASH server.
7. Restart DASH to allow the changes to take effect.

Customizing UI elements

You can customize a number of Agile Service Manager UI elements for your deployment, such as tooltips, link styles and icons. You can also create custom tools which users can access through a topology's context menu.

Tip: The Topology Viewer loads all UI configuration settings into memory when it opens. If you make changes to your UI configurations after opening the Topology Viewer, you must reopen it before these changes take effect.

Configuring custom tools

As an administrator or advanced user, you can create custom topology tools, which users can then access from within a topology's context menu. This functionality allows you to access properties of a selected item (such as a resource or relationship) and execute some custom functionality within the context of that item.

Before you begin

To access the Topology Tools page, you must have the admin role **inasm_admin** assigned to you. See the "Configuring DASH user roles" on page 21 topic for more information.

Note: You must be proficient in JavaScript to define custom tools.

About this task

Custom tools are written in JavaScript, and accessed through the right-click (context) menu in the UI. All inasm_operator users can access the tools, but only inasm_admin users can customize them.

Tip: The Refresh icon reloads the tools list. This can be useful if other users are customizing the tools.

Procedure

1. As the admin user, log into your DASH web application.
2. Select **Administration** from the DASH menu.
3. Select **Topology Tools** under the Agile Service Management heading.
4. Use the following information to complete the **Topology Tools - Details** page.

   Name Unique name used as an internal reference.

   Required.

   Menu label
   The menu label is the text displayed in the context menu.

   This can be the same name as used by other tools, which is why the unique name is required.

   Required

   Description
   A description to help administrator users record the tool's purpose.

   Not displayed in the context menu.

   Optional.

   Menu priority
   The menu priority slider defines where in the context menu the tool is displayed.

   For example, tools with a priority of two will be displayed higher in the menu than tools that have a priority of four.

   Available values are one to ten.

   Optional.

Navigation

You can move to the next page by using the page selector.

The minimum requirement to save the tool and open the Topology Tools - Implementation page is the name and label.

5. Use the following information to complete the **Topology Tools - Implementation** page. Here you create the JavaScript implementation for the tool, which defines the action that will occur when a user selects this option from the menu. An example JavaScript for a custom tool is included after these steps. To help you create custom Agile Service Manager tools, you have access to the following custom JavaScript helper functions:

   asmProperties

   The tool implementation has access to the properties of the relevant **resource**, **relationship** or **status** via the asmProperties JavaScript object, which contains all the properties.

   You can access the properties using standard JavaScript, but you must protect against a value not being present.

   For example if you intend to use the property 'latitude', you must verify that it is present before using it. To do so, use the following check command:

   ```
   asmProperties.hasOwnProperty('latitude')
   ```

   If the property is present, the Boolean value `true` will be returned.

   Status tools properties

   When creating **status** tools, you use JavaScript that is similar to the script that you use when creating **resource** or **relationship** tools. However, the properties you use in your status tool scripts, such as asmProperties, reference the properties for the **status** item; unlike the properties you use in your resource or relationship tool scripts, which reference the properties for the resources or relationships. For example, if you use asmProperties.location in a status tool script, there must be a corresponding 'location' property in the status record.

   When creating status tools, the asmProperties object has a property that takes the form of an array called **`resources`**, which represents the resources in the topology with which this status is associated. Each item in the resources array is an object with properties that represent the properties of that resource. For example, if a status is associated with two resources, the **`uniqueId`** property of the first of those two resources could be referenced in the script by using `asmProperties.resources[0].uniqueId`

   In addition, you can access the properties of a resource against which you are running a status tool by using the **`asmSourceProperties`** object when scripting the status tool.

   asmSourceProperties

   You can access information about the source properties of any **relationships** or **status** the custom tool is acting on via the asmSourceProperties JavaScript object.

   Example of using the source resource properties in a custom relationship stroke definition:

```
if (asmSourceProperties.myProp === 'high') {
    return 'blue';
} else {
    return 'black';
}
```

Remember: The arrows indicating a relationship point from the source to the target.

asmTargetProperties

You can access information about the target properties of **relationships** the custom tool is acting on via the asmTargetProperties JavaScript object.

asmFunctions

You can use a number of other helper functions, which are accessed from the asmFunctions object, which includes the following:

showConfirmationPopup(title, message, onOk)

Creates a popup confirmation allowing the tool to confirm an action.

Takes a title and message, which is displayed on the popup, and a function definition, which is run if the user clicks the OK button on the popup.

showToasterMessage(status, message)

Shows a popup toaster with the appropriate status coloring and message.

showPopup(title, text)

Shows a popup with a given title and text body, which can be generated based on the properties of the resource or relationship.

showIframe(url)

Displays a popup filling most of the page which wraps an iframe showing the page of the given URL.

Allows you to embed additional pages.

sendPortletEvent(event)

Allows you to send DASH portlet events from the Topology Viewer that can be used to manipulate other DASH portlets, such as the Event Viewer within IBM Tivoli Netcool/OMNIbus Web GUI.

Note: You can send events to other DASH portlets only if you are running Agile Service Manager within DASH (rather than in a direct-launch browser window), and if the receiving DASH portlets subscribe to the types of events being sent. See the “sendPortletEvent examples” on page 197 topic for more information.

asmFunctions.getResourceStatus(<resource_id>, <callback_function>, [<time_stamp>])

Allows you to request status information from a tool definition for a given resource using its **_id** parameter.

resource_id

Required

Can be obtained from a resource via `asmProperties._id` and from a relationship using `asmSourceProperties._id` or `asmTargetProperties._id`

callback_function
Required

Is called once the status data has been collected from the topology service, with a single argument containing an array of status objects

time_stamp
Optional

Unix millisecond timestamp to get the status from a given point in history

The following example prints the status information of a source resource from a relationship context to the browser console log:

```
let printStatusCallback = function(statuses) {
    statuses.forEach(function(status) {
        console.log('status:', status.status,
                    'state:', status.state,
                    'severity:', status.severity,
                    'time:', new Date(status.time));
    })
}
asmFunctions.getResourceStatus(asmSourceProperties._id,
printStatusCallback);
```

6. Use the following information to complete the **Topology Tools - Conditions** page. Here you define the resource or relationship conditions under which this tool is available.

Applicable item type for tool definition
From this drop-down, select the types to which the tool is applicable: **Resource**, **Relationship**, **Resource and Relationship**, or **Status**.

Depending on your selection, a number of check boxes are displayed, which you use to configure which resources, relationships or states are included.

All types / All states
Select this option if you want the tool to be displayed for all resource and relationship types, or all states (for Status).

The tool will also be displayed for any specific types not listed here.

Resource types
Select one or more resource types from the list displayed.

Relationship types
Select one or more relationship types from the list displayed.

Status Select from the following possible states for which the tool will be available:

- **Open**
- **Clear**
- **Closed**

Remember: When creating status tools, the properties you use in your status tool scripts reference the properties for the status item, while the

properties you use in your resource or relationship tools reference the properties for the resources or relationships.

Results

Once you have saved your changes, you must close the Topology Viewer and then reopen it to make the new tools available (tools will be available for use depending on the conditions set).

Example

The following example of a lookup tool is based on a 'person' resource with the properties 'name' and 'email'.

You first set the Topology Tools - Conditions page to use **Resource** as the applicable item type, and then select the **person** resource type only to ensure that the custom tool is only displayed for 'person' resources.

This example demonstrates the JavaScript for a tool that checks for properties, makes an HTTP request, and handles the response either by creating a popup with the response, or by showing an error message.

Note: The function `asmHelperFunctions.showToasterMessage` accepts the following values for the status:

- information
- normal
- warning
- escalated
- critical

```
// Check that the resource has the properties email and name
if(asmProperties.hasOwnProperty('email') && asmProperties.hasOwnProperty('name')){
    var emailAddress = asmProperties.email;
    var personName = asmProperties.name;

    // Build a http request to collect information from another service using the email
    var request = new XMLHttpRequest();
    var url = 'https://my-lookup-service?email=' + emailAddress;

    request.open('GET', url, true);

    request.onreadystatechange = function() {
        if (request.readyState === 4) {
            if (request.status === 200) {
                // On successful request show popup in UI window with custom title and
the json response from the request
                var title = personName + ' Details:';
                var content = JSON.stringify(JSON.parse(this.responseText), null, 4);
                asmFunctions.showPopup(title, content);
            } else {
               // On request error show popup in UI with custom message.
                var messageStatus = 'critical';
                var message = 'API error, '+ this.responseText
                asmFunctions.showToasterMessage(messageStatus, message);
            }
        }
    };

    request.send();
} else {

    // If resource doesn't have the expected properties, show toaster message with warning.
    var messageStatus = 'warning';
    var message = 'Unable to load service information, email address not provided.';
    asmFunctions.showToasterMessage(messageStatus, message);
}
```

sendPortletEvent examples

Using the Agile Service Manager custom tools functionality, you can send Agile Service Manager DASH portlet events to other DASH portlets, provided these subscribe to the types of events being sent. You use the sendPortletEvent helper function to define these portlet events. You can only send events to other DASH portlets if you are running Agile Service Manager within DASH, rather than in a direct-launch browser window.

Example 1: Topology Viewer and DASH Web Widget on same page

This event opens an internal personnel directory in the web widget for the clicked-on 'person' resource.

```
var address = 'https://adminsys:8080/staff?name=' + asmProperties.name;

var eventPayload = {
    'name': 'http://ibm.com/TIP#DisplayURL',
    'URL': address
};

asmFunctions.sendPortletEvent(eventPayload);
```

Example 2: Topology Viewer and Event Viewer on same page

This event updates the Netcool/OMNIbus Web GUI Event Viewer to display events where 'Node' matches the clicked resource name. It displays the Agile Service Manager Topology Viewer and Netcool/OMNIbus Event Viewer on the same page.

```
var whereClause = 'Node = \'' + asmProperties.name + '\'';

var eventPayload = {
   name: "http://ibm.com/tip#NodeClickedOn",
   payload: {
       product: {
           OMNIbusWebGUI: {
               displaycontext: {
                  "filterName": "HostEvents",
                  "filterType": "user_transient",
                  "registerFilter": "true",
                  "sql": whereClause,
                  "forceOverwrite": "true",
                  "viewName": "Default",
                  "viewType": "global",
                  "dataSource": "OMNIBUS"
                }
            }
        }
    }
};

asmFunctions.sendPortletEvent(eventPayload);
```

Example 3: Event Viewer on it own page

This event updates the Netcool/OMNIbus Web GUI Event Viewer to display events where 'Node' matches the clicked resource name. It displays the Netcool/OMNIbus Event Viewer on its own page.

```
var whereClause = 'Severity > 2 and Node = \'' + asmProperties.name + '\'';

var eventPayload = {
   name: "http://ibm.com/isclite#launchPage",
   NavigationNode: "item.desktop.navigationElement.EventViewer",
```

```
   switchPage: true,
   payload: {
      product: {
         OMNIbusWebGUI: {
            displaycontext: {
               "filterName": "HostEvents",
               "filterType": "user_transient",
               "registerFilter": "true",
               "sql": whereClause,
               "forceOverwrite": "true",
               "viewName": "Default",
               "viewType": "global",
               "dataSource": "OMNIBUS"
                }
            }
         }
    }
};

asmFunctions.sendPortletEvent(eventPayload);
```

Defining custom icons

You can add custom icons for resources that are displayed in the Agile Service Manager UI using the Custom Icons page accessed through DASH.

Before you begin

To access the Custom Icons page, you must have the admin role **inasm_admin** assigned to you. See the "Configuring DASH user roles" on page 21 topic for more information.

About this task

Tip: Instead of using the following procedure, you can first select a specific resource type from the Resource Types page, and then define a custom icon for that specific resource type only. This is described in step four "Editing resource type styles" on page 199. However, if you intend to create a number of icons without assigning them, or simply want to edit or delete icons, use the following procedure.

Icon properties

Each custom icon must have a name that uniquely identifies the icon when assigning it to a type.

Remember: You cannot change the name of an existing icon. If you want an icon to have a different name, create a new icon, then delete the old one.

Icons are defined inside an SVG editor, which performs an XML validation check.

Each icon definition must be valid svg xml with a given viewBox, which is important to ensure scaling of the image.

The svg definition must include inline styling of the image, such as stroke color and fill color. If style classes are used, naming must be unique for each svg image to prevent class definitions from being overwritten.

Optionally, each icon can be assigned to a category, which allows you to group icons of the same type or function together when displaying them in a list.

Procedure

1. As a user with the `inasm_admin` role, log into your DASH web application.
2. Select **Administration** from the DASH menu.
3. Select **Custom Icons** under the Agile Service Management heading.
4. Click **New** to create a new icon, and type a unique name. Alternatively, click the 'edit' symbol to edit an existing icon.

   Remember: You cannot change the name of an existing icon. If you want an icon to have a different name, create a new icon, then delete the old one.
5. Enter the SVG XML to define the icon. Use the editor to enter valid XML. The XML editor includes a Preview area where the results of your edits are displayed.

   Example: Use the following definition for the 'disk' icon as guidance:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64">
   <ellipse style="fill-opacity:0;stroke:currentColor;stroke-width:12.12270069;"
id="path4139" cx="33.627117" cy="32.949142" rx="16.803904" ry="17.210684"/>
   <circle cx="33.827423" cy="33.055576" r="3.3037829"/>
</svg>
```

6. Enter a category name, if required.
7. Click **Save**.

What to do next

Next, you assign these icons to particular resource types using the Resource Types page accessed through DASH. There, you can also apply further style edits to the resource types.

Related tasks:

"Editing resource type styles"
You assign existing or new custom icons to particular resource types using the Resource Types page accessed through DASH. Here you can also apply further resource type style edits, such as adding custom labels to a resource type, and changing its shape, size, border and background.

Editing resource type styles

You assign existing or new custom icons to particular resource types using the Resource Types page accessed through DASH. Here you can also apply further resource type style edits, such as adding custom labels to a resource type, and changing its shape, size, border and background.

Before you begin

To access the Custom Icons page, you must have the admin role **inasm_admin** assigned to you. See the "Configuring DASH user roles" on page 21 topic for more information.

About this task

Tip: If you intend to create a number of icons without assigning them to specific resource types, or simply want to edit or delete icons, use the procedure described in the "Defining custom icons" on page 198 topic. However, it may be more convenient to define custom icons as described from step four of the following procedure, as the custom icon is then immediately assigned to the previously selected resource type.

Procedure

1. As a user with the `inasm_admin` role, log into your DASH web application.
2. Select **Administration** from the DASH menu.
3. Select **Resource Types** under the Agile Service Management heading. The Resource Types page is displayed, which lists all existing resource types in table format in sortable columns. The table also displays the icons for the resource types and their names, categories (if defined), whether they are system icons or custom icons, and whether they have custom labels, styles or shapes. From here, you can delete resource types, but only if they are not yet being used in the topology data. You can also create new resource types, or edit existing ones, and then associate existing icons with them, and apply resource styling.
4. Click **New** to create a new resource type. Alternatively, click **Edit** to edit an existing resource type. The Configure Resource Type (or Edit Resource Type) page is displayed. You can toggle between the **Identification** and the **Styling** tabs.
5. On the **Identification** tab, define the selected resource type's icon, label and shape.
   a. For a new resource type, enter a unique name in the **Name** field. For an existing resource type, move on to the next step.

      Restriction: You cannot change the name of an existing resource type. If you want to delete an existing resource type, create a new one and assign an icon to it, and then delete the old one.

   b. Choose an icon to associate with the resource type using one of the following methods:

      From the Icon drop-down list
      If you open the **Icon** drop-down list, all icons are listed by name in alphabetical order.

      From the View all icons button
      If you click the **View all icons** button, all icons are displayed in alphabetical order.

      Click an icon to associate it with the resource type.

      From the Quick assign button
      If an icon exists with the same name as the resource type, click the **Quick assign** button to select it without having to sort through all existing icons.

      This function is useful if you have added a custom icon, and are now assigning it to a resource type with the same name.

      From the Define new custom icon button
      From here you can define a custom icon, which is automatically associated with the resource type when done.

      Click the **Define new custom icon** button to display the Custom Icons page.

      Click **New** to create a new icon. Alternatively, click the 'edit' symbol to edit an existing icon. After you have selected or created an icon, the Configure Resource Type page is displayed. Use the following information to define the icon:

Icon properties

Each custom icon must have a name that uniquely identifies the icon when assigning it to a type.

Remember: You cannot change the name of an existing icon. If you want an icon to have a different name, create a new icon, then delete the old one.

Icons are defined inside an SVG editor, which performs an XML validation check.

Each icon definition must be valid svg xml with a given viewBox, which is important to ensure scaling of the image.

The svg definition must include inline styling of the image, such as stroke color and fill color. If style classes are used, naming must be unique for each svg image to prevent class definitions from being overwritten.

Optionally, each icon can be assigned to a category, which allows you to group icons of the same type or function together when displaying them in a list.

Remember: You can also create custom icons from the Custom Icons page accessed through DASH, which is described in the "Defining custom icons" on page 198 topic.

c. Define the label for the resource type by editing the following fields:

Label By default the name of a property (`asmProperties.name`) is used as the label that is displayed in the topology.

You can replace this by typing in a custom label for the resource type.

Label Maximum Length

You can override the default label length of 15 characters to avoid truncating the displayed label.

Note: Avoid labels that are too long, as long labels may overlap in the topology. The maximum suggested label length is 20 characters.

d. Choose a shape for the resource type from the **Resource Shape** drop-down list. The default shape for a resource in the Topology Viewer is a circle, or a square for a host server. You can change the shape of the resource type to one of the following shapes:
   - Circle
   - Square
   - Hexagon
   - Vertical hexagon

6. On the **Styling** tab, define the selected resource type's border color, border pattern, background color, and resource display size.
   a. Change the border color by entering a hex definition. The default border color is `#152935` (a shade of black).
   b. Change the border pattern. The default pattern is `''`.
   c. Change the background color for the resource. The default is `#dfe3e6` (a very light blue).
   d. Select a size for the resource type. The default is medium. You can further refine the size function by specifying how certain resource properties effect

the size of the resource type displayed. For example, resource types can appear larger depending on the number of connections they have with other resources.

7. Click **Save** to return to the Resource Types page.

Results

The changes you have made to the resource type and its associated icon are now displayed in the Resource Types table.

Creating custom relationship type styles

You can customize the styles and labels for specified relationship types using the Relationship Types page accessed through DASH. You can also delete existing, or create new relationship type styles.

Before you begin

To access the Relationship Types, you must have the admin role **inasm_admin** assigned to you. See the "Configuring DASH user roles" on page 21 topic for more information.

About this task

Note: Any resource or relationship properties used in custom relationship styles must be added as required properties to the Agile Service Manager 'Global Settings' (**Administration** > **Global Settings**). However, if they do not exist in the Global Settings, you will be prompted to add them when you save your new relationship styles at the end of the following procedure.

You can set a relationship's line color, thickness, and style, as well as the label.

You customize styles in two steps. First, you identify the relationship type to be customized, or create a new one. Then you customize the style elements for the relationship type.

Default relationship style

When you customize relationship styles, you change the following default settings:

Line Color Function:

```
return '#1d3649';
```

Line Width Function:

```
return '1.5px';
```

Line Pattern Function:

```
return null;
```

Style restrictions

The line color (strokeWidth) property can have a maximum value of 9px, or it will cause styling problems.

The label (linkLabel) property must return a valid string, which will be displayed alongside the relationship.

See the following external sites for more detailed SVG style definition information:

Line color (stroke)
https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke

Line width (stroke-width)
https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/stroke-width

Line pattern (stroke-dasharray)
https://developer.mozilla.org/en/docs/Web/SVG/Attribute/stroke-dasharray

Accessing properties for styling functions
When defining styles you can access dynamic properties of the relationship or connected resources.

To access the properties of resources, use the asmProperties JavaScript object.

To access the properties of relationships, use the asmSourceProperties or asmTargetProperties JavaScript objects.

Remember: The arrows indicating a relationship point from the source to the target.

To access the highest open severity status from the source or target resource, use the asmSourceProperties._hasStatus or asmTargetProperties._hasStatus JavaScript objects. The following example uses the **_hasStatus** parameter to modify the relationship label:

```
if (asmSourceProperties._hasStatus || asmTargetProperties._hasStatus) {
    // object of all ASM's status severities
    let severityRank = {
        clear: 0,
        indeterminate: 1,
        information: 2,
        warning: 3,
        minor: 4,
        major: 5,
        critical: 6
    };
    let sourceSeverityRank = severityRank[asmSourceProperties._hasStatus] || 0;
    let targetSeverityRank = severityRank[asmTargetProperties._hasStatus] || 0;
    let labelString = asmProperties._edgeType;
    // Show highest Status on relationship label
    if(sourceSeverityRank > targetSeverityRank) {
        labelString += ': Status = ' + asmSourceProperties._hasStatus;
    } else {
        labelString += ': Status = ' + asmTargetProperties._hasStatus;
    }
    return labelString;
} else {
    // No status for source or target resources use plain label
    return asmProperties._edgeType;
}
```

Procedure

1. As a user with the `inasm_admin` role, log into your DASH web application.
2. Select **Administration** from the DASH menu.
3. Select **Relationship Types** under the Agile Service Management heading. The Relationship Types page is displayed, which lists all your existing customized relationship types in table format, also displaying a Last Updated time stamp, and whether a relationship type has a custom label, custom color or custom width defined. From here, you can delete or edit configurations. You can also create a new relationship type configuration.
4. Do one of the following:
   - To delete a relationship type, click the **Delete** icon.

- To edit an existing relationship type, click the **Edit** icon. The Configure Relationship Type page is displayed.
- To create a new relationship type, click **New**. The Configure Relationship Type page is displayed.

Configure relationship types

5. Select the **Identification** tab on the Configure Relationship Type page.

   Relationship Type
   Choose the relationship type that you want to configure from the dropdown list.

   Note: Only relationship types that exist in your topology database are listed.

   Label Function
   Example of a JavaScript function to define a label for the relationship using the resource properties:

   ```
   return asmProperties.labelText;
   ```

6. Select the **Style** tab on the Configure Relationship Type page.

   Line Color Function
   Example of a JavaScript function to define the ink color:

   ```
   return 'blue';
   ```

   Line Width Function
   Example of a JavaScript function to define the line width for the relationship using the source resource properties:

   ```
   return asmSourceProperties.myProp > 10 ? '9px' : '1.5px';
   ```

   Line Pattern Function
   Example of a JavaScript function to define the line pattern:

   ```
   return '3 4 5';
   ```

7. Click **Save**.

   Note: If any of the resource or link properties used have not yet been defined in global settings, you will be prompted to save them now.

Results

The relationship style and label for the specified relationship type has been customized.

Defining global settings

As a system administrator, you can define global settings, such as the URLs for trusted sites, the required properties for tooltips or relationship type styles, or the maximum hop numbers a user can choose in the Agile Service Manager topology viewer. You do this from the Global Settings page accessed through DASH.

Before you begin

To access the Global Settings page, you must have the admin role **inasm_admin** assigned to you. See the “Configuring DASH user roles” on page 21 topic for more information.

Important: Ensure that you understand your system's data capacity. If you allow users to set a high hop count, this will place greater demands on your network,

with more information being sent from the topology service to the topology viewer, and a greater workload on the topology service itself, and in the browser when rendering the topology.

About this task

Hop count

Choose a number between two and thirty.

The default maximum hop count is four.

Required properties

To improve system performance only specific resource and relationship properties are loaded into the Topology Viewer. You can specify additional properties to be fetched here.

These properties may then be used in tooltips, custom status tools or to customize UI elements (via custom JavaScript code).

Trusted sites

If you need to link to an HTTP address instead of HTTPS while writing a custom tool definition, and need to request data from a site that uses HTTP instead of HTTPS, then you can use the Agile Service Manager UI server as a proxy. The URL to the HTTP proxy takes the actual target HTTP site URL as a parameter. The proxy then sends the request itself, and returns the response back to the browser.

For security reasons, the HTTP proxy can only access the URLs that have been defined by an administrator user as 'trusted sites'.

The trustedSites property values are a comma-separated list of the trusted sites to which the proxy server can link, and from which it can retrieve information.

Trusted sites operate under a 'starts with' condition.

Procedure

1. As a user with the `inasm_admin` role, log into your DASH web application.
2. Select **Administration** from the DASH menu.
3. Select **Global Settings** under the Agile Service Management heading. The UI Global Settings page is displayed consisting of Topology Tools, Features and Rendering sections.
4. Select a value between two and thirty from the **Maximum number of hops allowed** drop-down list.

   CAUTION:
   Ensure that you set a maximum hop count that, if selected by an operator, will not overload your system's data capacity.
5. To add a required property, click the **Add** (+) button, and enter the property into the **Properties required for tooltips or relationship style** text box. For example: `location`

   To delete a property

   Select an existing property and click the **Delete** (-) button.
6. To add a trusted website, click the **Add** (+) button, and enter the required URLs into the **Trusted websites that can be accessed via HTTP** text box. For example, you could add the following trusted sites:

   ```
   www.ibm.com
   data-server.intranet.com:8080/network/statistics
   ```

In this example, the HTTP proxy will allow requests for all target URLs that start with 'www.ibm.com' or with 'data-server.intranet.com:8080/network/statistics', but no others, as illustrated in the following examples.

Allowed http proxy targets

```
www.ibm.com/cloud

data-server.intranet.com:8080/network/statistics/1893713/info
```

Not allowed http proxy targets

```
data-server.intranet.com:8080

www.ibm.co.uk

www.news-page.com
```

Configuring retention period for resource history

Agile Service Manager retains historical topology data for a default period of 30 days. You can increase this to a maximum of 90 days by increasing the 'time to live' (TTL) value.

Before you begin

CAUTION:
The performance and scalability of Agile Service Manager is affected by the number of resources managed, the amount of history present for each resource, and the ingestion rate. If you increase the retention period for historical resource data from the default of 30 days (up to a maximum of 90 days), your system performance (when rendering views) may deteriorate, if as a result of this increase resources have in excess of 25,000 historical entries.

About this task

The retention period for historical resource data is configured via the HISTORY_TTL configuration property.

When a resource or edge in Agile Service Manager is deleted, it will no longer appear in the UI. Historic representations of the resource or edge, however, are retained and can be accessed in the UI history timeline until the history TTL limit has been reached, after which the data is deleted.

Note: Unless explicitly deleted, live resources remain current and the TTL limit does not apply to them.

For an illustration. see the example section.

Procedure

Edit the configuration file

1. Open the `/opt/ibm/netcool/asm/.env` file using an appropriate editor.
2. Edit the HISTORY_TTL setting. Change the TTL value from the default of `30` to any value up to `90`.
3. Restart Agile Service Manager:

   ```
   $ASM_HOME/bin/asm_start.sh
   ```

Example

In the following example scenario, the default TTL value of 30 days applies.

Table 49. TTL example for the 'sprocket' resource

Date	Action	Topology view	Historical data
01-January-2019	**sprocket** resource created	**sprocket** resource visible in current and history	History is current resource
01-March-2019	**sprocket** resource modified	**sprocket** resource visible in current and history	History has both current and previous resource
31-March-2019	TTL expires for historic resource created on 01-March	**sprocket** resource visible in current and history	History is current (modified) resource only
01-May-2019	**sprocket** resource deleted	**sprocket** resource visible in history only	History still contains deleted resource
31-May-2019	TTL expires for historic resource created on 01-May	**sprocket** resource not visible in current or history	No history remains

Configuring the Helm chart to use alternate storage (ICP on OpenShift)

Agile Service Manager by default supports local volumes for storage on Kubernetes. To configure an alternate storage backend, you must set the storage class. This requires a manual command line installation, and **not** an installation via ICP UI. **Note** that this requirement is only likely when you are installing Agile Service Manager on IBM Cloud Private on OpenShift.

Before you begin

CAUTION:
You cannot configure storage class for installation from the ICP UI. This is an advanced command line installation that you should only perform if you have the required expertise.

About this task

Assumption:

- A suitable **`storageclass`** has been defined and configured to provision storage volumes
- For IBM Cloud Private on OpenShift installations only

Procedure

1. Find the storage class:

```
# kubectl get storageclass
NAME                    PROVISIONER                    AGE
vsphere                 kubernetes.io/vsphere-volume   7d18h
```

2. Set the storage class in the persistent volume claim:

   Important: The following example configuration snippet contains the relevant settings for storage class and volume capacity only. You must merge these settings with the other installation parameters.

```
global:
  persistence:
    enabled: true
    useDynamicProvisioning: true
    storageClassName: "vsphere"     #  to match value from 'kubectl get storageclass'
    storageClassOption:
      cassandradata: "default"
      zookeeperdata: "default"
      kafkadata: "default"
    storageSize:
      cassandradata: 50Gi
      kafkadata: 15Gi
      zookeeperdata: 5Gi
      elasticdata: 75Gi
```

Example

After you have changed the settings, the Agile Service Manager `PersistentVolumeClaims` will now include a storage class. On a system with an appropriate provisioner in place, the `PersistentVolumes` should be generated automatically, as in the following example:

```
NAME                                          READY   STATUS      RESTARTS   AGE
pod/asm-cassandra-0                           1/1     Running     0          10m
pod/asm-elasticsearch-0                       1/1     Running     0          10m
pod/asm-event-observer-58bdcd9d94-twj5q       1/1     Running     0          10m
pod/asm-kafka-0                               2/2     Running     0          10m
pod/asm-kubernetes-observer-698cfd746b-q7z9l  1/1     Running     0          10m
pod/asm-layout-84474476bc-96cz6               1/1     Running     0          10m
pod/asm-merge-c4cd8f8b7-nq589                 1/1     Running     0          10m
pod/asm-search-778b9f9574-hhsxf               1/1     Running     0          10m
pod/asm-system-health-cronjob-1560349200-qgkrr  0/1   Completed   0          9m10s
pod/asm-system-health-cronjob-1560349500-qqdnn  0/1   Completed   0          19s
pod/asm-topology-6b6c4b4b54-pp92j             1/1     Running     0          10m
pod/asm-ui-api-7849c55b4c-5qq2m               1/1     Running     0          10m
pod/asm-zookeeper-0                           1/1     Running     0          10m

NAME                                          STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/data-asm-cassandra-0  Bound    pvc-06784b19-8d1d-11e9-a948-005056b47772
50Gi RWO            vsphere        10m
persistentvolumeclaim/data-asm-elasticsearch-0   Bound    pvc-06f05e48-8d1d-11e9-a948-005056b47772
75Gi RWO            vsphere        10m
persistentvolumeclaim/data-asm-kafka-0           Bound    pvc-07e53665-8d1d-11e9-a948-005056b47772
15Gi RWO            vsphere        10m
persistentvolumeclaim/data-asm-zookeeper-0  Bound    pvc-085f9547-8d1d-11e9-a948-005056b47772
5Gi        RWO            vsphere        10m
```

Porting data for testing, backup and recovery

You can create backups of your Agile Service Manager UI configuration data in order to run a test configuration, or simply to safeguard your custom settings. You can also back up and restore your topology data.

Backing up and restoring database data (on-prem)

You can back up and restore existing topology data using scripts included in the Agile Service Manager **on-prem** installation. This can be helpful when updating your system, or for maintenance reasons.

About this task

The `backup_casssandra.sh` and `restore_cassandra.sh` scripts are stored in the `ASM_HOME/bin` directory.

Tip: The backup utility runs from inside the Cassandra Docker Container, and thus filesystem paths are relative from inside that container. Typically these file paths can be accessed from the host system via

`${ASM_HOME}/logs/cassandra`

and

`${ASM_HOME}/data/cassandra/backup`

Procedure

Backing up data

1. From the ASM_HOME directory, run the backup script: **Example**

```
# ./bin/backup_casssandra.sh

# Description : The backup script will complete the backup in multiple phases -
#  1. Take statistics of the keyspace(s) before backup
#  2. Clear existed snapshots
#  3. Take backup of keyspace(s) SCHEMA in temporary BACKUP_TEMP_DIR
#  4. Take snapshot of keyspace(s)
#  5. Copy snapshot to temporary BACKUP_TEMP_DIR
#  6. Compact the temporary BACKUP_TEMP_DIR in one tar file and send it to BACKUP_DIR

USAGE: backup_cassandra.sh
   [ -k keyspace to backup ] # default is ALL keyspaces
   [ -b  temporary backup dir ] # default is /opt/ibm/cassandra/data/backup/../backup_temp
   [ -d  datadir ] # default is /opt/ibm/cassandra/data/data
   [ -s storagedir ] # default is /opt/ibm/cassandra/data/backup
   [ -u Cassandra username ]
   [ -p Cassandra password ]
   [ -log logdir ] # default is /opt/ibm/cassandra/logs
   [ -speed tardiskspeed ] # default is 17M
   [ -f ] # for non interactive mode

********** START CONFIGURATION *****************
KEYSPACE_TO_BACKUP=ALL
BACKUP_TEMP_DIR=/opt/ibm/cassandra/data/backup/../backup_temp
BACKUP_DIR=/opt/ibm/cassandra/data/backup
CASSANDRA_DATA=/opt/ibm/cassandra/data/data
LOG_PATH=/opt/ibm/cassandra/logs
TAR_SPEED_LIMIT=17M
FORCE=N
USER=cassandra
PASS=XXXX
********** END CONFIGURATION *****************

Tue Mar  5 13:15:26 UTC 2019 ************************************
Tue Mar  5 13:15:26 UTC 2019  Do you want to continue (y/n) ?
```

Restore topology data

2. From the ASM_HOME directory, run the restore script: **Example**

```
# ./bin/restore_cassandra.sh

# Description : The restore script will complete the restore in multiple phases -
1. Take statistics of the cassandra node before restore
2. Check if the keyspace exists and if it does not exist, create it using the schema cql file
saved in the backup file
3. Truncate all tables in keyspace
4. Clear all files in commitlog directory
5. Copy contents of desired snapshot to active keyspace.
6. Refresh all tables in that keyspace
7. Take statistics of the cassandra node after restore and compare with statistics taken before
backup, making sure number of keys per table is the same

USAGE: restore_cassandra.sh
    -k keyspaceName  # compulsory parameter
   [ -h backup hostname] # if backup was done on a different hostname than 96c6953586a3
   [ -b temporary backup dir  ] # default is /opt/ibm/cassandra/data/backup/../backup_temp
   [ -d  dataDir ] # default is /opt/ibm/cassandra/data/data
   [ -t  snapshotTimestamp ] # timestamp of type date YYYY-MM-DD-HHMM-SS - default is latest
   [ -s  storageDir ] # default is /opt/ibm/cassandra/data/backup
   [ -u Cassandra username ]
   [ -p Cassandra password ]
   [ -log logDir ] # default is /opt/ibm/cassandra/logs
   [ -f ]  # for non interactive mode


********** START CONFIGURATION *****************
BACKUP_TEMP_DIR=/opt/ibm/cassandra/data/backup/../backup_temp
BACKUP_DIR=/opt/ibm/cassandra/data/backup
DATA_DIR=/opt/ibm/cassandra/data/data
LOG_PATH=/opt/ibm/cassandra/logs
LOCAL HOSTNAME=96c6953586a3
BACKUP_HOSTNAME=96c6953586a3
SNAPSHOT_DATE_TO_RESTORE=latest
KEYSPACE_TO_RESTORE=janusgraph
FORCE=N
USER=cassandra
```

```
PASS=XXXX
********** END CONFIGURATION ****************

Tue Mar  5 13:17:22 UTC 2019 ***********************************
Tue Mar  5 13:17:22 UTC 2019  Do you want to continue (y/n) ?
```

Backing up UI configuration data (on-prem)

Agile Service Manager includes a backup facility, which lets you backup UI configuration settings such as user preferences, topology tools, custom icons, relationship types, and global settings. This topic describes how to **export** these settings.

Before you begin

Remember: Backing up and restoring configuration is a two step process. The first step, exporting data, is described in this topic. The second step, importing data to restore previous configurations, is described in the "Restoring UI configuration data (on-prem)" on page 212 topic.

The tool detects the host, port, and tenant id for the Topology Service from the following environment variables:

- TOPOLOGY_SERVICE_HOST
- TOPOLOGY_SERVICE_PORT
- TENANT_ID

Important: If you have a standard installation of Agile Service Manager core, and none of the default settings have been changed, the tool will work without you having to reset any of these environment variables. However, if you do have a non-standard installation, you need to reset these before running the tool.

About this task

When to export configuration data

Agile Service Manager UI configuration settings are stored in the topology service database. If that database is deleted, the configuration settings are also deleted. You may therefore want to create a backup of configuration data if you intend to conduct testing that may involve changes to your database. After installing or rebuilding a new database, you can then restore the configuration data.

You can export configuration data as part of your data protection strategy, to provide a backup in case of data corruption or accidental data deletion.

You can export configuration data from a staging system in order to then import it into a live system.

Syntax

This command **must** be run from the ASM_HOME directory.

```
docker-compose -f tools.yml run --rm backup_ui_config [-config <config_type>]
[-out <output_filename>] [-force] [-verbose]
```

Tip: For help syntax:

```
docker-compose -f tools.yml run --rm backup_ui_config -help
```

Parameters

All parameters are optional.

Note: You can run the `backup_ui_config` command without setting any parameters. If you do, all Agile Service Manager UI configuration settings will be exported to the following default file: `$ASM_HOME/data/tools/asm_ui_config.txt`

config The `-config` flag allows the type of configuration you want to export to be specified.

By default all UI configuration settings are backed up.

Settings for `-config` <*config_type*>
Backs up the following Agile Service Manager UI configuration settings

all All UI configurations (default)

tools Topology tools definitions

icons Custom icon definitions

types Entity type definitions

links Relationship type definitions

preferences
User preferences

settings
Global settings

out The `-out` flag is the name of the backup file name to create.

The name must be a file name only, with no directory paths.

The default output file name is `asm_ui_config.txt`, and the output location is fixed as `$ASM_HOME/data/tools`.

Note: If the file already exists, the tool will indicate this and quit. For the existing file to be overwritten with new output, use the `-force` parameter.

force If you set the `-force` parameter, the tool overwrites an existing output file with new content.

verbose
The `-verbose` flag runs the tool in verbose mode, whereby extra log messages are printed to the shell during execution.

This parameter is useful if a problem occurs while running the tool, and you want to re-run it with extra information made available.

Procedure

1. Using the syntax information provided, determine the export (backup) `<options>` you need to set, if any.

   Remember: Run the `backup_ui_config` command without any options set to backup all UI configuration settings to `$ASM_HOME/data/tools/asm_ui_config.txt`.

2. Run the `backup_ui_config` command from your `ASM_HOME` directory, as in the following example. This runs the tool inside a docker container, and the `--rm` flag then causes the exited container to be deleted once the tool has completed.

   ```
   docker-compose -f tools.yml run --rm backup_ui_config <options>
   ```

Results

The Agile Service Manager UI configuration data is exported to the specified file in the `$ASM_HOME/data/tools` directory. If the `-force` flag has been set, an existing backup file is overwritten.

From the backup file, you can restore the settings using the "Restoring UI configuration data (on-prem)" topic.

Restoring UI configuration data (on-prem)

Agile Service Manager includes a backup facility, which lets you backup UI configuration settings such as user preferences, topology tools, custom icons, relationship types, and global settings. This topic describes how to **import** previously exported (backed up) settings in order to restore your previous configurations.

Before you begin

You can only import and restore configuration settings that have been previously exported, as described in the "Backing up UI configuration data (on-prem)" on page 210 topic

The tool detects the host, port, and tenant id for the Topology Service from the following environment variables:

- TOPOLOGY_SERVICE_HOST
- TOPOLOGY_SERVICE_PORT
- TENANT_ID

Important: If you have a standard installation of Agile Service Manager core, and none of the default settings have been changed, the tool will work without you having to reset any of these environment variables. However, if you do have a non-standard installation, you need to reset these before running the tool.

About this task

Syntax

This command **must** be run from the `ASM_HOME` directory.

```
docker-compose -f tools.yml run --rm import_ui_config -file <input_file>
[-overwrite] [-verbose]
```

Tip: For help syntax:

```
docker-compose -f tools.yml run --rm import_ui_config -help
```

Parameters

file The `-file` parameter is the name of the backup file from which to import definitions. It **must** be a file name only with no directory paths included, and it **must** exist in the tools data directory (`$ASM_HOME/data/tools`).

overwrite

By default, as the import tool reads the backup file it looks up each item in the topology service to see if it already exists. Any configuration definitions which already exist are **not** updated.

However, if you set the `-overwrite` flag, the existing definitions are overwritten with the values from the backup file.

verbose

The `-verbose` flag runs the tool in verbose mode, whereby extra log messages are printed to the shell during execution.

Useful if a problem occurs running the tool and you want to re-run it with extra information made available.

Procedure

1. Using the syntax information provided, enter the file name of the previously exported backup file.
2. Determine if any other `<options>` you need to set, such as the `-overwrite` flag.
3. Run the `import_ui_config` command from your `ASM_HOME` directory, as in the following example. This runs the tool inside a docker container, and the `--rm` flag then causes the exited container to be deleted once the tool has completed.

   ```
   docker-compose -f tools.yml run --rm import_ui_config <options>
   ```

Results

The Agile Service Manager UI configuration data is imported from the specified file in the `ASM_HOME/data/tools` directory. If the `-overwrite` flag has been set, existing configuration data will be overwritten.

Backing up database data (ICP)

You can back up (and later restore) existing topology data for the Agile Service Manager **ICP** installation. This can be helpful when updating your system, or for maintenance reasons.

About this task

To complete the backup, you complete a number of preparatory steps, then perform a data backup, and then restore the Agile Service Manager services.

Procedure

Preparing your system for backup

1. Authenticate into the Agile Service Manager Kubernetes namespace.
2. Deploy the kPodLoop bash shell function. **kPodLoop** is a bash shell function that allows a command to be run against matching Kubernetes containers. You can copy it into the shell.

   ```
   kPodLoop() {
    __podPattern=$1
    __podCommand=$2
    __podList=$( kubectl get pods --no-headers=true --output=custom-columns=NAME:.metadata.name |
   grep ${__podPattern} )
    printf "Pods found: $(echo -n ${__podList})\n"
    for pod in ${__podList}; do
       printf "\n===== EXECUTING COMMAND in pod: %-42s =====\n" ${pod}
       kubectl exec ${pod} -- bash -c "${__podCommand}"
       printf '_%.0s' {1..80}
       printf "\n"
    done;
   }
   ```

3. Make a note of the scaling of Agile Service Manager pods.

   ```
   kubectl get pods --no-headers=true --output=custom-columns=CNAME:.metadata.ownerReferences[0].name |
   grep asm | egrep -v -e 'noi|system-health' | uniq --count
   ```

   Example output:

   ```
   3 asm-cassandra
   1 asm-dns-observer
   3 asm-elasticsearch
   1 asm-event-observer
   1 asm-file-observer
   ```

```
3 asm-kafka
1 asm-kubernetes-observer
2 asm-layout
2 asm-merge
1 asm-rest-observer
2 asm-search
2 asm-topology
2 asm-ui-api
3 asm-zookeeper
```

4. Verify access to each Cassandra database (this command will return a list of keyspaces from each Cassandra node)

```
kPodLoop asm-cassandra "cqlsh -u \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -e
\"DESC KEYSPACES;\""
```

   The username and password variables are present in the container environment.

5. Suspend the system health (asm-system-health-cronjob) cronjob(s).

   Example 'list'

   ```
   kubectl get cronjobs.batch
   ```

   Example 'edit'

   ```
   kubectl edit cronjobs.batch asm-system-health-cronjob
   ```

   This opens a yaml file in vi.

   a. Locate:
      `suspend: false`
   b. Change to:
      `suspend: true`
   c. Save the yaml file.

   Alternatively, run the following command:

   ```
   $ kubectl patch cronjobs.batch/asm-system-health-cronjob -p '{"spec":{"suspend":true}}'
   ```

   Example 'check'

   ```
   kubectl get cronjobs.batch asm-system-health-cronjob
   ```

6. Scale down Agile Service Manager pods.

```
kubectl scale deployment --replicas=0 asm-dns-observer
kubectl scale deployment --replicas=0 asm-event-observer
kubectl scale deployment --replicas=0 asm-file-observer
kubectl scale deployment --replicas=0 asm-rest-observer
kubectl scale deployment --replicas=0 asm-kubernetes-observer
kubectl scale deployment --replicas=0 asm-layout
kubectl scale deployment --replicas=0 asm-merge
kubectl scale deployment --replicas=0 asm-search
kubectl scale deployment --replicas=0 asm-ui-api
kubectl scale deployment --replicas=0 asm-topology
```

7. Verify.

```
kubectl get pods | grep asm | grep -v noi
```

   Note: The asm-cassandra, asm-elasticsearch, asm-kafka and asm-zookeeper pods will remain active.

Backing up data

8. Deploy the `pbkc` bash shell function. The `pbkc` function attempts to backup the Cassandra database on all nodes as close to simultaneously as possible. You can copy it into the shell.

```
pbkc() {
 ## Parallel Backup of Kubernetes Cassandra
 DATE=$( date +"%F-%H-%M-%S" )
 LOGFILEBASE=/tmp/clusteredCassandraBackup-${DATE}-
 declare -A PIDWAIT
 declare -A LOG

 ## get the current list of asm-cassandra pods.
 podlist=$( kubectl get pods --no-headers=true --output=custom-columns=NAME:.metadata.name | grep asm-cassandra )
 for pod in ${podlist}; do
    LOG[$pod]=${LOGFILEBASE}${pod}.log
    echo -e "BACKING UP CASSANDRA IN POD ${pod} (logged to ${LOG[$pod]})"
    kubectl exec ${pod} -- bash -c "/opt/ibm/backup_scripts/backup_cassandra.sh  -u \${CASSANDRA_USER} -p
\${CASSANDRA_PASS} -f" > ${LOG[$pod]} & PIDWAIT[$pod]=$!
 done
```

```
 echo -e "${#PIDWAIT[@]} Backups Active ..."

 for pod in ${podlist}; do
    wait ${PIDWAIT[$pod]}
    echo -e "Backup of ${pod} completed, please verify via log file (${LOG[$pod]})"
 done
}
```

9. Run a clean-up on all keyspaces in all Cassandra instances. Example Cassandra keyspaces cleanup:

```
kPodLoop asm-cassandra "nodetool cleanup system_schema"
kPodLoop asm-cassandra "nodetool cleanup system"
kPodLoop asm-cassandra "nodetool cleanup system_distributed"
kPodLoop asm-cassandra "nodetool cleanup system_auth"
kPodLoop asm-cassandra "nodetool cleanup janusgraph"
kPodLoop asm-cassandra "nodetool cleanup system_traces"
```

10. Run backup on all Cassandra instances (using the `pbkc` shell function just deployed).

```
pbkc
```

11. Check the final output in the log file for each backup. Adjust the date in the grep command as appropriate.

```
grep  "BACKUP DONE SUCCESSFULLY" /tmp/clusteredCassandraBackup-2019-06-14-14-09-50*
/tmp/clusteredCassandraBackup-2019-06-14-14-09-50-asm-cassandra-0.log:Fri Jun 14 14:11:04 UTC 2019 BACKUP DONE
SUCCESSFULLY !!!
/tmp/clusteredCassandraBackup-2019-06-14-14-09-50-asm-cassandra-1.log:Fri Jun 14 14:11:16 UTC 2019 BACKUP DONE
SUCCESSFULLY !!!
/tmp/clusteredCassandraBackup-2019-06-14-14-09-50-asm-cassandra-2.log:Fri Jun 14 14:11:16 UTC 2019 BACKUP DONE
SUCCESSFULLY !!!
```

Restore services

12. Enable the system health (asm-system-health-cronjob) cronjob(s).

    Example 'list'

    ```
    kubectl get cronjobs.batch
    ```

    Example 'edit'

    ```
    kubectl edit cronjobs.batch asm-system-health-cronjob
    ```

    This opens a yaml file in vi.

    a. Locate:
       `suspend: true`
    b. Change to:
       `suspend: false`
    c. Save the yaml file.

    Alternatively, run the following command:

    ```
    $ kubectl patch cronjobs.batch/asm-system-health-cronjob -p '{"spec":{"suspend":false}}'
    ```

    Example 'check'

    ```
    kubectl get cronjobs.batch asm-system-health-cronjob
    ```

13. Scale up the services to the original level. The original level was obtained in a previous step.

```
kubectl scale deployment --replicas=3 asm-topology
kubectl scale deployment --replicas=2 asm-layout
kubectl scale deployment --replicas=2 asm-merge
kubectl scale deployment --replicas=2 asm-search
kubectl scale deployment --replicas=2 asm-ui-api
kubectl scale deployment --replicas=1 asm-dns-observer
kubectl scale deployment --replicas=1 asm-event-observer
kubectl scale deployment --replicas=1 asm-file-observer
kubectl scale deployment --replicas=1 asm-rest-observer
kubectl scale deployment --replicas=1 asm-kubernetes-observer
```

What to do next

You can restore your backed up data as and when required.

Restoring database data (ICP)

You can restore existing topology data for the Agile Service Manager **ICP** installation, if backed up earlier. This can be helpful when updating your system, or for maintenance reasons.

About this task

To complete the restoration of your data, you complete a number of preparatory steps, then perform a data restore, and then restore the Agile Service Manager services.

Procedure

Preparing your system for data restoration

1. Authenticate into the Agile Service Manager Kubernetes namespace.
2. Deploy the kPodLoop bash shell function. **kPodLoop** is a bash shell function that allows a command to be run against matching Kubernetes containers. You can copy it into the shell.

```
kPodLoop() {
 __podPattern=$1
 __podCommand=$2
 __podList=$( kubectl get pods --no-headers=true --output=custom-columns=NAME:.metadata.name |
grep ${__podPattern} )
 printf "Pods found: $(echo -n ${__podList})\n"
 for pod in ${__podList}; do
    printf "\n===== EXECUTING COMMAND in pod: %-42s =====\n" ${pod}
    kubectl exec ${pod} -- bash -c "${__podCommand}"
    printf '_%.0s' {1..80}
    printf "\n"
 done;
}
```

3. Make a note of the scaling of Agile Service Manager pods.

```
kubectl get pods --no-headers=true --output=custom-columns=CNAME:.metadata.ownerReferences[0].name |
grep asm | egrep -v -e 'noi|system-health' | uniq --count
```

Example output:

```
3 asm-cassandra
1 asm-dns-observer
3 asm-elasticsearch
1 asm-event-observer
1 asm-file-observer
3 asm-kafka
1 asm-kubernetes-observer
2 asm-layout
2 asm-merge
1 asm-rest-observer
2 asm-search
2 asm-topology
2 asm-ui-api
3 asm-zookeeper
```

4. Verify access to each Cassandra database (this command will return a list of keyspaces from each Cassandra node)

```
kPodLoop asm-cassandra "cqlsh -u \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -e
\"DESC KEYSPACES;\""
```

5. Suspend the system health (asm-system-health-cronjob) cronjob(s).

Example 'list'

```
kubectl get cronjobs.batch
```

Example 'edit'

```
kubectl edit cronjobs.batch asm-system-health-cronjob
```

This opens a yaml file in vi.

a. Locate:
   `suspend: false`
b. Change to:
   `suspend: true`

c. Save the yaml file.

Alternatively, run the following command:

```
$ kubectl patch cronjobs.batch/asm-system-health-cronjob -p '{"spec":{"suspend":true}}'
```

Example 'check'

```
kubectl get cronjobs.batch asm-system-health-cronjob
```

6. Scale down Agile Service Manager pods.

```
kubectl scale deployment --replicas=0 asm-dns-observer
kubectl scale deployment --replicas=0 asm-event-observer
kubectl scale deployment --replicas=0 asm-file-observer
kubectl scale deployment --replicas=0 asm-rest-observer
kubectl scale deployment --replicas=0 asm-kubernetes-observer
kubectl scale deployment --replicas=0 asm-layout
kubectl scale deployment --replicas=0 asm-merge
kubectl scale deployment --replicas=0 asm-search
kubectl scale deployment --replicas=0 asm-ui-api
kubectl scale deployment --replicas=0 asm-topology
```

7. Verify.

```
kubectl get pods | grep asm | grep -v noi
```

Note: The asm-cassandra, asm-elasticsearch, asm-kafka and asm-zookeeper pods will remain active.

Restore data

8. Update the Cassandra restore script to suppress the truncation of restored data.

Note: The `restore_cassandra.sh` tool truncates all data in the target table each time it is used, and despite the restore being targeted at one Cassandra node only, the truncate is propagated to all nodes. In order to suppress the truncate step, you must update the restore script on all but the first node.

a. Copy `cassandra_functions.sh` out of one of the asm-cassandra nodes.

```
kubectl cp asm-cassandra-0:/opt/ibm/backup_scripts/cassandra_functions.sh /tmp/.
```

b. Edit cassandra_functions.sh

```
vi /tmp/cassandra_functions.sh
```

Locate the call to `truncate_all_tables` within the `restore()` function and comment out the appropriate lines, as in the following example:

```
Printf "`date` Starting Restore \n"

#### truncate_all_tables
#### testResult $? "truncate tables"

repair_keyspace
```

c. Save the file, then copy the file back to all nodes, except the first Cassandra node.

```
kubectl cp cassandra_functions.sh asm-cassandra-2:/opt/ibm/backup_scripts/cassandra_functions.sh
 kubectl cp cassandra_functions.sh asm-cassandra-1:/opt/ibm/backup_scripts/cassandra_functions.sh
```

9. Locate the timestamps of the backups from each Cassandra node to restore. Each node's backup was started at a similar time, so the timestamps may differ by a few seconds. In the following example a backup was performed at about 2019-06-11 09:36, and grep is then used to filter to these backup archives:

```
kPodLoop asm-cassandra "ls -larth \${CASSANDRA_DATA}/../backup_tar | grep 2019-06-11-09"
Pods found: asm-cassandra-0 asm-cassandra-1 asm-cassandra-2

===== EXECUTING COMMAND in pod: asm-cassandra-0                          =====
-rwxrwxr-x 1 cassandra cassandra 524M Jun 11 09:37 cassandra_asm-cassandra-0_KS_system_schema_KS_system_
KS_system_distributed_KS_system_auth_KS_janusgraph_KS_system_traces_date_2019-06-11-0936-04.tar
________________________________________________________________________________

===== EXECUTING COMMAND in pod: asm-cassandra-1                          =====
-rwxrwxr-x 1 cassandra cassandra 565M Jun 11 09:37 cassandra_asm-cassandra-1_KS_system_schema_KS_system_
KS_system_distributed_KS_system_auth_KS_janusgraph_KS_system_traces_date_2019-06-11-0936-07.tar
________________________________________________________________________________

===== EXECUTING COMMAND in pod: asm-cassandra-2                          =====
```

```
-rwxrwxr-x 1 cassandra cassandra 567M Jun 11 09:37 cassandra_asm-cassandra-2_KS_system_schema_KS_system_
KS_system_distributed_KS_system_auth_KS_janusgraph_KS_system_traces_date_2019-06-11-0936-07.tar
_____________________________________________________________________
```

Tip: You can ignore this step if you are about to apply the most recent backup. If you do, the **-t** parameter can be omitted during all subsequent steps.

10. Working across each Cassandra node, restore the relevant backup of the `system_auth` keyspace. While this updates the credentials, it is also important to run the `nodetool repair` after the restore to each node.
    a. `asm-cassandra-0`

       Remember: This will cause the existing data in the `system_auth` keyspace tables to be truncated.

       ```
       kPodLoop asm-cassandra-0 "/opt/ibm/backup_scripts/restore_cassandra.sh -k system_auth -t 2019-06-11-0936-04 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-0 "nodetool repair --full system_auth"
       ```

    b. `asm-cassandra-1`

       ```
       kPodLoop asm-cassandra-1 "/opt/ibm/backup_scripts/restore_cassandra.sh -k system_auth -t 2019-06-11-0936-07 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-0 "nodetool repair --full system_auth"
       ```

    c. `asm-cassandra-2`

       ```
       kPodLoop asm-cassandra-2 "/opt/ibm/backup_scripts/restore_cassandra.sh -k system_auth -t 2019-06-11-0936-07 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-0 "nodetool repair --full system_auth"
       ```

11. Working across each Cassandra node, restore the relevant backup of the `janusgraph` keyspace. While this updates the credentials, it is also important to run the `nodetool repair` after the restore to each node.
    a. `asm-cassandra-0`

       Remember: This will cause the existing data in the `janusgraph` keyspace tables to be truncated.

       ```
       kPodLoop asm-cassandra-0 "/opt/ibm/backup_scripts/restore_cassandra.sh -k janusgraph  -t 2019-06-11-0936-04 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-0 "nodetool repair --full janusgraph"
       ```

    b. `asm-cassandra-1`

       ```
       kPodLoop asm-cassandra-1 "/opt/ibm/backup_scripts/restore_cassandra.sh -k janusgraph  -t 2019-06-11-0936-07 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-1 "nodetool repair --full janusgraph"
       ```

    c. `asm-cassandra-2`

       ```
       kPodLoop asm-cassandra-2 "/opt/ibm/backup_scripts/restore_cassandra.sh -k janusgraph  -t 2019-06-11-0936-07 -u
       \${CASSANDRA_USER} -p \${CASSANDRA_PASS} -f"
       kPodLoop asm-cassandra-2 "nodetool repair --full janusgraph"
       ```

Restore services

12. Enable the system health (asm-system-health-cronjob) cronjob(s).

    Example 'list'

    ```
    kubectl get cronjobs.batch
    ```

    Example 'edit'

    ```
    kubectl edit cronjobs.batch asm-system-health-cronjob
    ```

    This opens a yaml file in vi.

    a. Locate:
       `suspend: true`
    b. Change to:
       `suspend: false`
    c. Save the yaml file.

    Alternatively, run the following command:

    ```
    $ kubectl patch cronjobs.batch/asm-system-health-cronjob -p '{"spec":{"suspend":false}}'
    ```

Example 'check'

```
kubectl get cronjobs.batch asm-system-health-cronjob
```

13. Scale up the services to the original level.

```
kubectl scale deployment --replicas=3 asm-topology
kubectl scale deployment --replicas=2 asm-layout
kubectl scale deployment --replicas=2 asm-merge
kubectl scale deployment --replicas=2 asm-search
kubectl scale deployment --replicas=2 asm-ui-api
kubectl scale deployment --replicas=1 asm-dns-observer
kubectl scale deployment --replicas=1 asm-event-observer
kubectl scale deployment --replicas=1 asm-file-observer
kubectl scale deployment --replicas=1 asm-rest-observer
kubectl scale deployment --replicas=1 asm-kubernetes-observer
```

14. Rebroadcast data to ElasticSearch (that is, re-index Elasticsearch).

    If data in Elasticsearch is out of sync with data in the Cassandra database, resynchronize it by calling the `rebroadcast` API of the topology service. This triggers the rebroadcast of all known resources on Kafka, and the Search service will then index those resources in Elasticsearch.

    Workaround

    Call the `rebroadcast` API of the Topology service, specifying a tenantId:

    `https://`*`master_fqdn`*`/1.0/topology/swagger#!/Crawlers/rebroadcastTopology`

Backing up and restoring UI configuration data (ICP)

Agile Service Manager on ICP includes a backup facility, which lets you backup UI configuration settings such as user preferences, topology tools, custom icons, relationship types, and global settings.

Procedure

1. Find the name of the topology pod, as in the following example:

```
$ kubectl get pod --namespace default --selector app=topology
NAME                              READY     STATUS    RESTARTS   AGE
asm-topology-577dc5497b-2wbxk   1/1       Running   0          12h
```

2. Run the backup tool using `kubectl exec`, as in the following examples:
   Example A

```
$ kubectl exec -ti asm-topology-577dc5497b-2wbxk -- /opt/ibm/graph.tools/bin/backup_ui_config -help

 usage: backup_ui_config [-config <config_type>] [-out <output_filename>] [-force] [-verbose]

 where 'config-type' can be set to one of the following:

 all         - backup all ASM UI configuration (default)
 tools       - backup topology tools definitions
 icons       - backup custom icon definitions
 types       - backup entity type definitions
 links       - backup relationship type definitions
 preferences - backup user preferences
 settings    - backup global settings
```

   Example B

```
$ kubectl exec -ti asm-topology-577dc5497b-2wbxk --
/opt/ibm/graph.tools/bin/backup_ui_config -out backup-20180908.json
 INFO    : Topology Service REST host detected: localhost:8080
 INFO    : Topology Service tenant ID detected: cfd95b7e-3bc7-4006-a4a8-a73a79c71255
 WARNING : No topology tool definitions were found
 WARNING : No custom icon definitions were found
 INFO    : Backing up entity type: container
 INFO    : Backing up entity type: cpu
 INFO    : Backing up entity type: deployment
 INFO    : Backing up entity type: image
 INFO    : Backing up entity type: namespace
 INFO    : Backing up entity type: namespace
 INFO    : Backing up entity type: networkinterface
 INFO    : Backing up entity type: operatingsystem
 INFO    : Backing up entity type: pod
 INFO    : Backing up entity type: server
 INFO    : Backing up entity type: service
 INFO    : Backing up entity type: volume
```

```
 WARNING : No relationship type definitions were found
 WARNING : No user preferences definitions were found
 WARNING : No global settings definitions were found
 INFO    : Output file has been created: /opt/ibm/netcool/asm/data/tools/backup-20180908.json

 Program complete.
```

3. Run the import tool, as in the following example:

```
$ kubectl exec -ti asm-topology-577dc5497b-2wbxk --
/opt/ibm/graph.tools/bin/import_ui_config -file backup-20180908.json -overwrite
 INFO  : Topology Service REST host detected: localhost:8080
 INFO  : Topology Service tenant ID detected: cfd95b7e-3bc7-4006-a4a8-a73a79c71255
 INFO  : Skipping import of entity type because it matches the existing definition: container
 INFO  : Skipping import of entity type because it matches the existing definition: cpu
 INFO  : Skipping import of entity type because it matches the existing definition: deployment
 INFO  : Skipping import of entity type because it matches the existing definition: image
 INFO  : Skipping import of entity type because it matches the existing definition: networkinterface
 INFO  : Skipping import of entity type because it matches the existing definition: psu
 INFO  : Skipping import of entity type because it matches the existing definition: router
 INFO  : Skipping import of entity type because it matches the existing definition: sensor
 INFO  : Skipping import of entity type because it matches the existing definition: server
 INFO  : Skipping import of entity type because it matches the existing definition: service
 INFO  : Skipping import of entity type because it matches the existing definition: subnet
 INFO  : Skipping import of entity type because it matches the existing definition: switch
 INFO  : Skipping import of entity type because it matches the existing definition: vlan
 INFO  : Skipping import of entity type because it matches the existing definition: vpn

 Program complete.
```

4. To save a copy of your backup, copy the file out of the topology container using the `kubectl cp` command. For example:

```
$ kubectl cp asm-topology-577dc5497b-2wbxk:/opt/ibm/netcool/asm/data/tools/backup-20180908.json
/tmp/backup-20180809.json
 $ find /tmp/backup*
 /tmp/backup-20180809.json
```

5. To import files, copy them into the `/opt/ibm/netcool/asm/data/tools` location inside the container:

```
$ kubectl cp /tmp/backup-20180809.json asm-topology-577dc5497b-2wbxk:/opt/ibm/netcool/asm/data/
tools/backup-20180909.json
 $ kubectl exec -ti asm-topology-577dc5497b-2wbxk -- find /opt/ibm/netcool/asm/data/tools/
 /opt/ibm/netcool/asm/data/tools/backup-20180908.json
 /opt/ibm/netcool/asm/data/tools/backup-20180909.json
```

Launching in context from OMNIbus Event Viewer

You can set up launch-in-context functionality in DASH from a Netcool/OMNIbus Web GUI event list (Event Viewer) to an Agile Service Manager Topology Viewer portlet, using a DASH **`NodeClickedOn`** action event.

Using the NodeClickedOn DASH event to launch the topology viewer with a set of parameters is similar to using a direct-launch URL (as described in the following topic: "Accessing topologies via direct-launch URL string" on page 164).

Advantages of using a NodeClickedOn event for launch-in-context (rather than a direct URL)

Removes the need to re-load the entire Topology Viewer when rendering a new topology.

Allows the use of the `asmFunctions.sendPortletEvent` function to the opened Topology Viewer.

Updating a topology on the same DASH page

You can add a right-click menu item to the Netcool/OMNIbus Web GUI event list, which you can use to update an already-open Agile Service Manager Topology Viewer that is on the **same** page in DASH as the event list.

Procedure

1. As a user with the Web GUI admin role, log into your DASH web application.
2. Open the Netcool/OMNIbus Web GUI tool configuration page: **Administration** > **Event Management Tools** > **Tool Configuration**
3. Create a new Script tool, with a script command similar to the following example:

   In this example the topology seed is a vertex name whose value is derived from the `Node` field in a Netcool/OMNIbus event.

   ```
   var eventPayload = {
       "name": "http://ibm.com/tip#NodeClickedOn",
       "payload": {
           "product": {
               "AgileServiceManager": {
                   "resourceName": "{@Node}",
                   "hops": 2,
                   "hopType": "e2h"
               }
           }
       }
   };
   {$param.portletNamespace}sendPortletEvent(eventPayload);
   ```

   Tip: Use the following list of supported parameters: Table 50 on page 222
4. Open the Netcool/OMNIbus Web GUI menu configuration page: **Administration** > **Event Management Tools** > **Menu Configuration**
5. Add the tool to the **Alerts** menu.
6. In DASH, click **Console Settings** > **General** > **Pages** and create a new page.
7. Add an Event Viewer portlet and a Topology Viewer portlet to the page, and arrange them as required.

Results

When you select an event from your Event Viewer portlet and launch your new tool, the Topology Viewer on the same DASH page will be updated, and render the topology for the seed whose resource `name` property matches the `Node` field value from the selected event.

Updating a topology on a different DASH page

You can add a right-click menu item to the Netcool/OMNIbus Web GUI event list, which you can use to update a Topology Viewer that is on a **different** DASH page from the event list.

Procedure

1. As a user with the Web GUI admin role, log into your DASH web application.
2. Open the Netcool/OMNIbus Web GUI tool configuration page: **Administration** > **Event Management Tools** > **Tool Configuration**
3. Create a new Script tool, with a script command similar to the following example:

The 'NavigationNode' value must be the Page Unique Name for the DASH page that you want to launch. In this example it is the unique name of the out-of-the-box Topology Viewer page.

```
var eventPayload = {
    "name": "http://ibm.com/isclite#launchPage",
    "NavigationNode": "web.netcool.asm.topologyViewer.page",
    "switchPage": "true",
    "payload": {
        "product": {
            "AgileServiceManager": {
                "resourceName": "{@Node}",
                "hops": "3",
                "layoutType": "4",
                "layoutOrientation": "TopToBottom"
            }
        }
    }
};
{$param.portletNamespace}sendPortletEvent(eventPayload);
```

Tip: Use the following list of supported parameters: Table 50

4. Open the Netcool/OMNIbus Web GUI menu configuration page: **Administration** > **Event Management Tools** > **Menu Configuration**
5. Add the tool to the **Alerts** menu.

Results

When you select an event from your Event Viewer portlet and launch your new tool, the Topology Viewer page will be opened or updated, and render the topology for the seed whose resource `name` property matches the `Node` field value from the selected event.

Launch-in-context parameters

The event payload of a DASH event used to update a Topology Viewer can have any of the parameters listed in this topic.

Parameters

Tip: The following parameters are also supported by the direct-launch URL facility.

Table 50. Launch-in-context parameters

Parameter	Type	Purpose
deltaTime	Integer	A point in history to compare to, as a unixtime
hideSearch	String	Hides the top search bar if set to 'true'
hideToolbar	String	Hides the side toolbar if set to 'true'
hops	Integer	The number of hops from the seed to display
hopType	String	The type of hops to display, for example 'host', 'e2h'
layoutOrientation	String	The layout orientation, for example 'TopToBottom'
layoutType	Integer	The topology layout type, as a number

Table 50. Launch-in-context parameters (continued)

Parameter	Type	Purpose
refreshTime	Integer	The refresh rate in milliseconds (when **not** in historical mode)
resourceId	String	The '_id' of the seed resource
resourceName	String	The 'name' of the seed resource
resourceUniqueId	String	The 'uniqueId' of the seed resource
time	Integer	The point in history to view the topology, as a unixtime

Defining rules

Rules help streamline topologies and conserve system resources, for example by merging different observer records of the same resource into a single composite resource, or by excluding specific changes from being recorded against a resource history.

Before you begin

- You must know your tenant ID.
- You also need to know specific details about resources for which you intend to develop rules. For example, to create merge rules you must know which resources exist as duplicate records before you can merge these into composites.

Version 1.1.5 Notice:

Existing Agile Service Manager 1.1.4 merge rules will work without the need of any migration, but any scripts that contain rules need to be changed to the new Agile Service Manager 1.1.5 format if they are to be run on Version 1.1.5 (or later).

About this task

You can use a number of different types of rules for different purposes. The rule type (**ruleType**) can be one the following:

mergeRule
: A merge rule populates the tokens of resources matched by the rule to prevent duplicate records of the same resource from being displayed in the topology.

  See About Merge Rules for more information.

tagsRule
: A tags rule populates the **tags** of resources matched by the rule.

matchTokensRule
: A match token rule populates the **matchTokens** of resources matched by the rule.

historyRule
: A history rule allows you to exclude properties from being retained in history, thereby saving resources by not maintaining detailed historical records of changes to these properties.

  See About History Rules for more information.

About Merge Rules:

Different observers deployed as part of the Agile Service Manager solution may record and then display the same resource as two (or more) resources. To prevent this, you create a merge rule that ensures that the separate records of the same resource share values in their tokens set, which then triggers the Merge Service to create a single composite resource vertex. Composite, merged resources are displayed in the Topology Viewer as a single resource, which includes the properties of all merged resources.

Merge rules are applied to a resource in a observer job before it is sent to the topology service. Rules can be managed using the Rules REST API in the Merge Service. For each Agile Service Manager observer, merge rules control which tokens are considered merge tokens. Live Swagger documentation for merge rules is here:

`http://<your host>/1.0/merge/swagger/#/Rules`

The following example is the default merge rule defined for the Docker Observer:

```
rules:
    - name: dockerId
      ruleType: mergeRule
      ruleStatus: enabled
      mergeTokens: [ dockerId ]
      entityTypes: null
      observers: [ docker-observer ]
      providers: null
    - name: almExternalId
      ruleType: mergeRule
      ruleStatus: enabled
      mergeTokens: [ externalId ]
      entityTypes: null
      observers: [ alm-observer ]
      providers: null
```

Notice that the rules name in this example is **dockerId**, and that it applies only to instances of observers named **docker-observer**. The ruleType property here specifies the **mergeRule** rule type. This merge rule applies to all entity types and all providers and will copy the **dockerId** property into the merge tokens for all resources that have this property set.

About History Rules:

A history rule populates the **historyExcludeTokens** field with field names.

When a resource is updated, the topology service checks if **all** the updates are on fields that are listed in **historyExcludeTokens**, and if they are, it updates the resource **without** creating history.

Procedure

Using the live Swagger interface or cURL commands, write a rule for each record, and POST it to the Rules API.
Use the following information as guidance when defining each rule:

Name The name of the rule, which must be unique within the context of the tenant.

Rule type (ruleType)
The rule type specifies the type of rule, and can be one the following:

- **mergeRule**
- **tagsRule**
- **matchTokensRule**
- **historyRule**

Status (ruleStatus)
The rule status can be either enabled or disabled, and the observers will only apply rules which are in an enabled state.

Merge tokens (mergeTokens)
The tokens set in a merge rule contains the list of resource parameter names which will become merge tokens for those resources to which the rule is applied.

Merge tokens can be constructed using variable substitutions, which allows you to combine more than one property value in a token, and also combine them with literal strings, as shown in this example.

Important: The tokens are the shared elements that the duplicate records to be merged have in common.

Entity types (entityTypes)
The entity types set in a rule contain the list of resource entity types for which this rule is valid.

If omitted, the rule will apply to all entity types.

Observers (observers)
The observers set contains the list of names of the observers to which this rule applies.

If omitted, or set to include the value '*', the rule will apply to all observers.

Providers (providers)
The providers set contains the list of names of the providers so which this rule applies.

If omitted, or set to include the value '*', the rule will apply to all providers.

You can use the mutually exclusive **excludeTokens** and **includeTokens** properties to filter providers.

Exclude tokens (excludeTokens)
These properties discard any values that match the regular expression.

Include tokens (includeTokens)
These properties apply a token only if the value matches the regular expression.

Example of tagsRule and matchTokensRule:

```
- name: matchRule
    ruleType: matchTokensRule
    ruleStatus: enabled
    mergeTokens: [ name ]
    entityTypes: null
    observers: null
    providers: null
  - name: tagRuleCustomProp
    ruleType: tagsRule
    ruleStatus: enabled
```

```
mergeTokens: [ name ]
entityTypes: null
observers: null
providers: null
```

Results

After you have created and posted your rules, these are applied before each observer job is sent to the topology service.

Example

Sample rules for merging resources
A composite resource has its own unique identifier, and the merged resources continue to exist separately. The following example shows the individual and composite resources when retrieved from the topology **Resources** API.

Resource one

http://<*your_NASM_host*>/1.0/topology/resources/ABC

```
{
  "_id": "ABC",
  "name": "host1",
  "propertyAbc": "This property only exists on ABC",
  "entityTypes": [ "host" ],
  "mergeTokens": [ "host1MergeToken" ],
  "_compositeId": "XYZ"
}
```

The resource has an id of ABC, and the value of compositeId denotes the single, merged resource, which is XYZ.

Resource two

http://<*your host*>/1.0/topology/resources/DEF

```
{
  "_id": "DEF",
  "name": "host1",
  "propertyDef": "This property only exists on DEF",
  "entityTypes": [ "host" ],
  "mergeTokens": [ "host1MergeToken" ],
  "_compositeId": "XYZ"
}
```

The resource has an id of DEF, and the value of compositeId denotes the single, merged resource, which is XYZ.

Composite resource

http://<*your_NASM_host*>/1.0/topology/resources/XYZ

```
{
  "_id": "XYZ",
  "name": "host1",
  "propertyAbc": "This property only exists on ABC",
  "propertyDef": "This property only exists on DEF",
  "entityTypes": [ "host" ],
  "mergeTokens": [ "host1MergeToken" ],
  "_compositeOfIds": [ "ABC", "DEF " ]
}
```

The resource has an id of XYZ, and the value of compositeOfIds lists the ids of the merged resources, in this case ABC and DEF. The XYZ composite resource includes the properties from both of the merged resources.

Resource with variable substitutions and exclude list

```
{
    "name": "sysNameMatching",
    "mergeTokens": [ "sysName", "${name}/${customField}"],
    "ruleStatus": "enabled",
    "entityTypes": [ "host", "server" ],
    "observers": [ "ITNM", "TADDM" ],
    "providers": [ "*" ],
    "customField": "string",
    "excludeTokens": [ "^asm-default.*"]
}
```

The `^asm-default.*` value set for excludeTokens ensures that any values that match the regular expressions are excluded.

The merge token with the value of `${name}/${customField}` combine the `${name}` and `${customField}` properties using the `${}` syntax, and demonstrate how variable substitutions work.

- Literal values are entered as they are in the merge token, which in this case is the / character.
- To be backwards compatible, tokens consisting of a single value, as in the `sysName` example, are treated as variable substitutions, that is, as if they are `${sysName}`.

Tip: You can also view composite resources in the live Swagger Composite API in the Merge Service, which returns the properties of the composite itself, and provides methods to get all composite vertices: `http://<your host>/1.0/merge/swagger/#/Composites`

Improving database performance

You can improve the performance of Agile Service Manager, such as fine-tuning Cassandra database cluster operations.

Changing the Cassandra `gc_grace_seconds` value (ICP)

The Cassandra database cluster performance may be impacted by tombstone occurrences which results in slower query response times. When this performance degrades the Agile Service Manager installation, the following procedure to set `gc_grace_seconds` can be used to mitigate this degradation.

About this task

For an ICP environment with a Cassandra cluster, `gc_grace_seconds` can be reduced from the default value of 864000 seconds (10 days).

This parameter impacts the ability of the Cassandra cluster to repair itself after a node has been offline.

Define a value for `gc_grace_seconds` that is greater than the duration of any anticipated Cassandra node outage.

Procedure

1. Find the name of a Cassandra pod. The change can be carried out on any node as the change will be replicated across the nodes.

   ```
   kubectl get pods | grep cass
   ```

   For example:

```
$ kubectl get pods | grep cass
asm-cassandra-0     1/1     Running     0     9d
```

The pod is identified as asm-cassandra-0

2. Run the following command to exec into the pod and start **cqlsh**:

```
kubectl exec -ti {pod name} -- cqlsh -u cassandra -p cassandra
```

For example:

```
$ kubectl exec -ti asm-cassandra-0 -- cqlsh -u cassandra -p cassandra
Connected to apm_cassandra at asm-cassandra-0:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh>
```

Repeat steps 3 to 5 for all tables within the janusgraph key space.

3. Verify the current setting of gc_grace_seconds.

```
SELECT table_name,gc_grace_seconds FROM system_schema.tables
WHERE keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,gc_grace_seconds
FROM system_schema.tables WHERE keyspace_name='janusgraph';

 table_name              | gc_grace_seconds
-------------------------+------------------
               edgestore |           864000
         edgestore_lock_ |           864000
              graphindex |           864000
        graphindex_lock_ |           864000
          janusgraph_ids |           864000
       system_properties |           864000
 system_properties_lock_ |           864000
               systemlog |           864000
                   txlog |           864000
```

4. Update the values using the **ALTER TABLE** command:

```
ALTER TABLE janusgraph.{table name} WITH gc_grace_seconds = {gc_grace_seconds value};
```

For example:

```
cassandra@cqlsh> ALTER TABLE janusgraph.edgestore WITH gc_grace_seconds = 345600;
```

5. Verify the settings have worked.

```
SELECT table_name,gc_grace_seconds FROM system_schema.tables
WHERE keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,gc_grace_seconds
FROM system_schema.tables WHERE keyspace_name='janusgraph';

 table_name              | gc_grace_seconds
-------------------------+------------------
               edgestore |           345600
         edgestore_lock_ |           864000
              graphindex |           864000
        graphindex_lock_ |           864000
          janusgraph_ids |           864000
       system_properties |           864000
 system_properties_lock_ |           864000
               systemlog |           864000
                   txlog |           864000

(9 rows)
```

Remember: Repeat steps 3 to 5 for all tables within the janusgraph key space.

6. Exit **cqlsh**: **Example:**

```
cassandra@cqlsh> exit
$
```

Changing the Cassandra gc_grace_seconds value (on-prem)

The Cassandra database cluster performance may be impacted by tombstone occurrences which results in slower query response times. When this performance degrades the Agile Service Manager installation, the following procedure to set gc_grace_seconds can be used to mitigate this degradation.

About this task

For an on-prem environment, gc_grace_seconds can be safely set to 0.

Procedure

1. Log into a server where Agile Service Manager is running.
2. Find the name of a Cassandra container.

```
docker ps | grep cassandra
```

For example:

```
$ docker ps | grep cassandra
000000000000   nasm-cassandra:3.11.3.62   "/opt/ibm/start-ca..."
5 hours ago   Up 5 hours   asm_cassandra_1
```

The pod is identified as asm-cassandra-1

3. Run the following command to exec into the container and start **cqlsh**:

```
docker exec -ti {container name} cqlsh -u cassandra -p cassandra
```

For example:

```
$ docker exec -ti asm_cassandra_1 cqlsh -u cassandra -p
Connected to topology_cassandra at asm_cassandra_1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh>
```

Repeat steps 4 to 6 for all tables within the janusgraph key space.

4. Verify the current setting of gc_grace_seconds.

```
SELECT table_name,gc_grace_seconds FROM system_schema.tables
WHERE keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,gc_grace_seconds
FROM system_schema.tables WHERE keyspace_name='janusgraph';

 table_name              | gc_grace_seconds
-------------------------+------------------
                edgestore |           864000
          edgestore_lock_ |           864000
               graphindex |           864000
         graphindex_lock_ |           864000
           janusgraph_ids |           864000
        system_properties |           864000
 system_properties_lock_  |           864000
                systemlog |           864000
                    txlog |           864000
```

5. Change the value to 0 (zero) using the **ALTER TABLE** command:

```
ALTER TABLE janusgraph.{table name} WITH gc_grace_seconds = {gc_grace_seconds value};
```

For example:

```
cassandra@cqlsh> ALTER TABLE janusgraph.edgestore WITH gc_grace_seconds = 0;
```

6. Verify the settings have worked.

```
SELECT table_name,gc_grace_seconds FROM system_schema.tables WHERE
keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,gc_grace_seconds FROM system_schema.tables
WHERE keyspace_name='janusgraph';

 table_name              | gc_grace_seconds
-------------------------+------------------
               edgestore |                0
         edgestore_lock_ |           864000
              graphindex |           864000
        graphindex_lock_ |           864000
          janusgraph_ids |           864000
       system_properties |           864000
 system_properties_lock_ |           864000
               systemlog |           864000
                   txlog |           864000

(9 rows)
```

Remember: Repeat steps 4 to 6 for all tables within the janusgraph key space.

7. Exit **cqlsh**: **Example:**

```
cassandra@cqlsh> exit
$
```

Changing the Cassandra dclocal_read_repair_chance value (ICP)

In ICP environments, the Cassandra cluster performance can be improved by setting dclocal_read_repair_chance to 0, thereby removing this Cassandra functionality. This functionality is not required as consistency issues are resolved by all Agile Service Manager 'read' and 'write' activities using a consistency level of QUORUM.

Procedure

1. Find the name of a Cassandra pod. The change can be carried out on any node as the change will be replicated across the nodes.

```
kubectl get pods | grep cass
```

For example:

```
$ kubectl get pods | grep cass
asm-cassandra-0     1/1     Running     0     9d
```

The pod is identified as asm-cassandra-0

2. Run the following command to exec into the pod and start **cqlsh**:

```
kubectl exec -ti {pod name} -- cqlsh -u cassandra -p cassandra
```

For example:

```
$ kubectl exec -ti asm-cassandra-0 -- cqlsh -u cassandra -p cassandra
Connected to apm_cassandra at asm-cassandra-0:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh>
```

Repeat steps 4 to 6 for all tables within the janusgraph key space.

3. Verify the current setting of dclocal_read_repair_chance.

```
SELECT table_name,dclocal_read_repair_chance FROM system_schema.tables
WHERE keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,dclocal_read_repair_chance
FROM system_schema.tables WHERE keyspace_name='janusgraph';

 table_name              | dclocal_read_repair_chance
-------------------------+----------------------------
               edgestore |                        0.1
        edgestore_lock_  |                        0.1
             graphindex  |                        0.1
       graphindex_lock_  |                        0.1
         janusgraph_ids  |                        0.1
      system_properties  |                        0.1
 system_properties_lock_ |                        0.1
               systemlog |                        0.1
                   txlog |                        0.1

(9 rows)
```

4. Update the value to 0 (zero) using the **ALTER TABLE** command:

```
cassandra@cqlsh> ALTER TABLE janusgraph.edgestore WITH dclocal_read_repair_chance = 0;
```

5. Verify the change has worked.

```
SELECT table_name,gc_grace_seconds,dclocal_read_repair_chance FROM
system_schema.tables WHERE keyspace_name='janusgraph';
```

For example:

```
cassandra@cqlsh> SELECT table_name,gc_grace_seconds,dclocal_read_repair_chance
FROM system_schema.tables WHERE keyspace_name='janusgraph';

  table_name              | dclocal_read_repair_chance
-------------------------+----------------------------
               edgestore |                          0
        edgestore_lock_  |                        0.1
             graphindex  |                        0.1
       graphindex_lock_  |                        0.1
         janusgraph_ids  |                        0.1
      system_properties  |                        0.1
 system_properties_lock_ |                        0.1
               systemlog |                        0.1
                   txlog |                        0.1

(9 rows)
```

Remember: Repeat steps 4 to 6 for all tables within the janusgraph key space.

6. Exit **cqlsh**: **Example:**

```
cassandra@cqlsh> exit
$
```

Configuring scaling for ICP

You can scale your Agile Service Manager deployment vertically or horizontally, and you can scale out as well as in.

Remember:

scaling
You can scale up you system horizontally or vertically.

Horizontal scaling means increasing the replication factor of a particular service, and may also require adding additional hardware.

Vertical scaling means that you add more power (CPU or RAM) to an existing machine.

Important: To avoid data loss, enable persistence before scaling up your system.

Tip: The redistribution process places additional load on the cluster, so should be performed at quiet times.

Restriction: Scaling for Agile Service Manager observers is **not** supported, and neither is scaling from a single to multiple instances.

Scaling vertically

To scale up Agile Service Manager vertically, you increase available CPU or RAM resources.

About this task

The amount of CPU and RAM requested by Agile Service Manager services is controlled by the configuration option **`global.environmentSize`** with valid values being:

size0 Specifies the least amount of resources required to run Agile Service Manager.

Recommended for testing or proof-of-concept deployments only, and not suitable for high availability (HA) mode.

size1 Specifies the resource requirements that allow Agile Service Manager to run production workloads.

Suitable for HA mode.

See the "Sizing reference" on page 313 topic for more details.

Scaling horizontally

To scale up Agile Service Manager horizontally, you first provision additional persistent volumes, then scale the component, and then redistribute, reassign or repair data across all nodes.

The following Agile Service Manager components can be scaled horizontally by adding additional machines, or pods, to your deployment:

- **Cassandra** database
- **ElasticSearch** search and analytics engine
- **Kafka** message bus

- **Zookeeper** synchronization service

Assumption: This task and the examples provided assume that you have deployed a standard production environment with three instances of each of the components that are to be scaled horizontally.

Scaling horizontally: Persistence

Before you scale any of the components, you provision new storage volumes for the additional replicas to use.

Before you begin

Source the kubhelper.sh helper function

The kubhelper.sh is provided in the Agile Service Manager `pak_extensions` directory.

Source the helper script as follows:

```
$ source pak_extensions/common/kubhelper.sh
```

About this task

The helper function has the following parameters:

- **Kubernetes worker node** for the volume
- **Helm release name** containing the application that needs a volume
- **Kubernetes namespace** where application is to be installed
- **Claim name** that needs storage
- **Storage capacity** required
- **Path** on the worker that will be used for storage

Procedure

Example: The following example adds storage for an additional three Kafka brokers.

1. Three brokers have already been installed on the first three worker nodes.

```
$ kubectl get pod -l app=kafka,release=asm -o wide
NAME          READY   STATUS    RESTARTS   AGE    IP            NODE             NOMINATED NODE   READINESS GATES
asm-kafka-0   2/2     Running   0          108m   10.1.205.80   172.16.188.122   <none>           <none>
asm-kafka-1   2/2     Running   0          108m   10.1.34.201   172.16.154.233   <none>           <none>
asm-kafka-2   2/2     Running   0          108m   10.1.91.94    172.16.183.205   <none>           <none>
```

2. Considering the three brokers on the first three worker nodes, create volumes on other hosts so that the Kafka cluster remains resilient to node failures. These are the other nodes:

```
$ kubectl get nodes
NAME            STATUS   ROLES                              AGE   VERSION
172.16.153.121   Ready    etcd,management,master,proxy,va   35h   v1.13.5+icp-ee
172.16.154.233   Ready    worker                             35h   v1.13.5+icp-ee
172.16.183.205   Ready    worker                             35h   v1.13.5+icp-ee
172.16.188.122   Ready    worker                             35h   v1.13.5+icp-ee
172.16.190.218   Ready    worker                             35h   v1.13.5+icp-ee
172.16.191.196   Ready    worker                             35h   v1.13.5+icp-ee
172.16.192.211   Ready    worker                             35h   v1.13.5+icp-ee
172.16.192.70    Ready    worker                             35h   v1.13.5+icp-ee
```

3. Choose the next three workers in the list (that is, `172.16.190.218`, `172.16.191.196` and `172.16.192.211`) and then create the volumes as follows:

```
$ createPersistentVolume 172.16.190.218 asm netcool data-asm-kafka-3 15 /opt/ibm/netcool/asm/data/kafka
Wed Jun 12 13:35:26 PDT 2019  INFO: Checking if '172.16.190.218' is a valid worker node - OK
Wed Jun 12 13:35:26 PDT 2019  INFO: Checking if 'netcool' is a valid namespace - OK
Wed Jun 12 13:35:26 PDT 2019  INFO: Creating volume for pvc 'netcool/data-asm-kafka-3' with capacity '15Gi'
at path '/opt/ibm/netcool/asm/data/kafka' on node '172.16.190.218'
persistentvolume/172.16.190.218-data-asm-kafka-3 created
```

```
$ createPersistentVolume 172.16.191.196 asm netcool data-asm-kafka-4 15 /opt/ibm/netcool/asm/data/kafka
Wed Jun 12 13:35:56 PDT 2019  INFO: Checking if '172.16.191.196' is a valid worker node - OK
Wed Jun 12 13:35:57 PDT 2019  INFO: Checking if 'netcool' is a valid namespace - OK
Wed Jun 12 13:35:57 PDT 2019  INFO: Creating volume for pvc 'netcool/data-asm-kafka-4' with capacity '15Gi'
at path '/opt/ibm/netcool/asm/data/kafka' on node '172.16.191.196'
persistentvolume/172.16.191.196-data-asm-kafka-4 created

$ createPersistentVolume 172.16.192.211 asm netcool data-asm-kafka-5 15 /opt/ibm/netcool/asm/data/kafka
Wed Jun 12 13:36:38 PDT 2019  INFO: Checking if '172.16.192.211' is a valid worker node - OK
Wed Jun 12 13:36:38 PDT 2019  INFO: Checking if 'netcool' is a valid namespace - OK
Wed Jun 12 13:36:38 PDT 2019  INFO: Creating volume for pvc 'netcool/data-asm-kafka-5' with capacity '15Gi'
at path '/opt/ibm/netcool/asm/data/kafka' on node '172.16.192.211'
persistentvolume/172.16.192.211-data-asm-kafka-5 created
```

4. Finally, create the paths on these workers.

```
$ ssh root@172.16.190.218 mkdir -p /opt/ibm/netcool/asm/data/kafka
$ ssh root@172.16.191.196 mkdir -p /opt/ibm/netcool/asm/data/kafka
$ ssh root@172.16.192.211 mkdir -p /opt/ibm/netcool/asm/data/kafka
```

Scaling horizontally: Cassandra database

This task described how to scale the Cassandra database (both out and in).

Before you begin

A default production deployment will have a replication factor of three. That means that with three nodes, each one should contain a complete copy of the data, as shown in this example:

```
nodetool status for asm-cassandra-0
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load        Tokens       Owns (effective)  Host ID                               Rack
UN  10.1.91.112  126.21 MiB  256          100.0%            dec6be10-4dcc-493e-8c26-330c19a32da2  rack1
UN  10.1.34.216  126.18 MiB  256          100.0%            e79c86a8-2105-4fad-b2ab-d10cdcd8354d  rack1
UN  10.1.205.91  126.12 MiB  256          100.0%            77851686-5715-4237-86e4-31b62603ac2b  rack1
```

About this task

Scale out

To scale out and spread the data load, you add additional nodes to the Cassandra cluster.

Scale in

To scale in a Cassandra cluster, you must first decommission nodes, starting with the highest numbered pod.

During this process data must be moved to the remaining cluster nodes. The decommission process instructs the node being decommissioned to move its data elsewhere (which is essentially the opposite of bootstrapping).

Remember: This process places additional load on the cluster, so ideally needs to be performed at quiet times.

Procedure

Scale out

1. Provision extra storage. Before you scale Cassandra, you will need to provision extra storage, as also described in more detail in the “Scaling horizontally: Persistence” on page 233 topic. You must provision the additional storage on worker nodes other than the current ones being used, so that the Cassandra cluster remains resilient to node failures, as in the following example:

```
$ source pak_extensions/common/kubhelper.sh
$ createPersistentVolume 172.16.190.218 asm netcool data-asm-cassandra-3 50 /opt/ibm/netcool/asm/data/cassandra
$ createPersistentVolume 172.16.191.196 asm netcool data-asm-cassandra-4 50 /opt/ibm/netcool/asm/data/cassandra
$ ssh root@172.16.190.218 mkdir -p /opt/ibm/netcool/asm/data/cassandra
$ ssh root@172.16.191.196 mkdir -p /opt/ibm/netcool/asm/data/cassandra
```

2. Update deployment configuration. Update the installation configuration with the desired number of Cassandra nodes in the cluster. For clarity, only the Cassandra cluster size is shown here.

```
global:
  cassandraNodeReplicas: 5
```

3. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

4. Once the upgrade completes, check that the additional Cassandra pods are ready.

```
$ watch kubectl get pod -lapp=cassandra,release=asm
NAME              READY   STATUS    RESTARTS   AGE
asm-cassandra-0   1/1     Running   1          4d21h
asm-cassandra-1   1/1     Running   2          4d21h
asm-cassandra-2   1/1     Running   1          4d21h
asm-cassandra-3   1/1     Running   0          3m36s
asm-cassandra-4   1/1     Running   0          3m36s
```

5. Verify that the data is now distributed across all nodes. Check the cluster 'nodetool' status according to each node:

```
for pod in `kubectl get pod -l app=cassandra,release=asm | grep -v NAME | awk '{print $1}'`; do
   echo "nodetool status for $pod"
   kubectl exec $pod /opt/ibm/cassandra/bin/nodetool status
   echo
done
```

   Scaled out to to five nodes, the three copies should be distributed across five nodes with each having about 60% of the data:

```
nodetool status for asm-cassandra-4
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load        Tokens       Owns (effective)  Host ID                               Rack
UN  10.1.87.160  72.4 MiB    256          59.8%             d89e123f-3e98-401f-8442-d5521ad50daf  rack1
UN  10.1.91.112  132.22 MiB  256          56.4%             dec6be10-4dcc-493e-8c26-330c19a32da2  rack1
UN  10.1.34.216  132.33 MiB  256          63.0%             e79c86a8-2105-4fad-b2ab-d10cdcd8354d  rack1
UN  10.1.205.91  132.21 MiB  256          59.1%             77851686-5715-4237-86e4-31b62603ac2b  rack1
UN  10.1.54.122  103.34 MiB  256          61.7%             0a99bbb8-3383-4cd0-b3c6-87d9f3909981  rack1
```

Scale in

In the following example the use of five nodes is scaled back to three nodes.

6. Decommission the highest numbered node. You go into the fifth container, check the current status of the cluster and then start the decommission process, which moves the data elsewhere.

```
$ kubectl exec -ti asm-cassandra-4 bash
[cassandra@asm-cassandra-4 /]$ /opt/ibm/cassandra/bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load        Tokens       Owns (effective)  Host ID                               Rack
UN  10.1.34.242  155.64 MiB  256          63.0%             e79c86a8-2105-4fad-b2ab-d10cdcd8354d  rack1
UN  10.1.87.166  100.22 MiB  256          59.8%             d89e123f-3e98-401f-8442-d5521ad50daf  rack1
UN  10.1.91.70   155.57 MiB  256          56.4%             dec6be10-4dcc-493e-8c26-330c19a32da2  rack1
UN  10.1.205.93  147.69 MiB  256          59.1%             77851686-5715-4237-86e4-31b62603ac2b  rack1
UN  10.1.54.124  128.39 MiB  256          61.7%             0a99bbb8-3383-4cd0-b3c6-87d9f3909981  rack1

[cassandra@asm-cassandra-4 /]$ /opt/ibm/cassandra/bin/nodetool decommission
[cassandra@asm-cassandra-4 /]$
```

7. After the decommission process completes, check the cluster status. This should show that five nodes have been reduced to four, each now having approximately 75% of the data.

```
$ /opt/ibm/cassandra/bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load        Tokens       Owns (effective)  Host ID                               Rack
UN  10.1.34.242  173.04 MiB  256          80.1%             e79c86a8-2105-4fad-b2ab-d10cdcd8354d  rack1
UN  10.1.91.70   155.54 MiB  256          71.4%             dec6be10-4dcc-493e-8c26-330c19a32da2  rack1
UN  10.1.205.93  147.69 MiB  256          73.1%             77851686-5715-4237-86e4-31b62603ac2b  rack1
UN  10.1.54.124  147.99 MiB  256          75.4%             0a99bbb8-3383-4cd0-b3c6-87d9f3909981  rack1
```

8. Repeat the process for the fourth Cassandra node.

After completion, there should be three nodes, each with 100% of the data:

```
[cassandra@asm-cassandra-3 /]$ /opt/ibm/cassandra/bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load        Tokens       Owns (effective)  Host ID                               Rack
UN  10.1.34.242  203.76 MiB  256          100.0%            e79c86a8-2105-4fad-b2ab-d10cdcd8354d  rack1
UN  10.1.91.70   167.36 MiB  256          100.0%            dec6be10-4dcc-493e-8c26-330c19a32da2  rack1
UN  10.1.205.93  161.14 MiB  256          100.0%            77851686-5715-4237-86e4-31b62603ac2b  rack1
```

9. Update the deployment configuration to remove the additional pods:

```
global:
  cassandraNodeReplicas: 3
```

10. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

11. Check that the additional pods are stopped, and the cluster status of the remaining nodes.
12. Deprovision the additional storage.
    - Clean up the persistent volumes, starting with the claims first:

      ```
      $ kubectl delete pvc data-asm-cassandra-3 data-asm-cassandra-4
      persistentvolumeclaim "data-asm-cassandra-3" deleted
      persistentvolumeclaim "data-asm-cassandra-4" deleted
      ```

    - Check that the persistent volumes are released:

      ```
      ibmadmin@asm-prod-master:~$ kubectl get pv | grep cass
      NAME                                        CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM         STORAGECLASS  REASON  AGE
      172.16.154.233-data-asm-cassandra-0   50Gi      RWO           Retain          Bound     netcool/data-asm-cassandra-0        7d2h
      172.16.183.205-data-asm-cassandra-1   50Gi      RWO           Retain          Bound     netcool/data-asm-cassandra-1        7d2h
      172.16.188.122-data-asm-cassandra-2   50Gi      RWO           Retain          Bound     netcool/data-asm-cassandra-2        7d2h
      172.16.190.218-data-asm-cassandra-3   50Gi      RWO           Retain          Released  netcool/data-asm-cassandra-3        45h
      172.16.191.196-data-asm-cassandra-4   50Gi      RWO           Retain          Released  netcool/data-asm-cassandra-4        45h
      ```

    - Delete the now redundant persistent volumes:

      ```
      $ kubectl delete pv 172.16.190.218-data-asm-cassandra-3  172.16.191.196-data-asm-cassandra-4
      persistentvolume "172.16.190.218-data-asm-cassandra-3" deleted
      persistentvolume "172.16.191.196-data-asm-cassandra-4" deleted
      ```

    - Clean the actual volumes on the worker nodes:

      ```
      $ ssh root@172.16.190.218 rm -rf /opt/ibm/netcool/asm/data/cassandra
      $ ssh root@172.16.191.196 rm -rf /opt/ibm/netcool/asm/data/cassandra
      ```

    Note: Always make absolutely sure that you are cleaning the correct nodes.

Scaling horizontally: Elasticsearch search and analytics engine

Elasticsearch data is stored in an index split into a number of shards, which distribute data around a cluster. To achieve high availability, these shards are replicated and distributed across the cluster.

Before you begin

The Agile Service Manager Search service by default creates a number of indices, which essentially are current resources and historical resources. The default number of shards is five, which you can customize using the **ELASTICSEARCH_SHARDS** variable.

About this task

The number of replica shards depends on how many Elasticsearch nodes there are, but there would never be more than two replicas. When there are three Elasticsearch nodes, an index has five shards and there are two replicas of those shards, meaning there are three copies of the data spread over three nodes. One of those copies is elected the Primary.

Replicas are used to provide redundant copies of your data to protect against hardware failure, and to serve read requests, like searching or retrieving a document.

Scale out

To scale out Elasticsearch, you provision extra storage, update the deployment configuration, and then perform a helm upgrade.

You can verify that the scale out was successful by checking cluster status and health, as well as node and shard health.

Scale in

To avoid data loss, you scale in Elasticsearch one node at a time.

Elasticsearch automatically redistributes shards, but you must wait for the scale in of each node to succeed before proceeding to scale in the next node.

Procedure

Scale out

1. Provision extra storage. Before you scale Elasticsearch, you will need to provision extra storage, as also described in more detail in the "Scaling horizontally: Persistence" on page 233 topic. You must provision the additional storage on worker nodes other than the current ones being used, so that the Elasticsearch cluster remains resilient to node failures, as in the following example:

```
$ source pak_extensions/common/kubhelper.sh
$ createPersistentVolume 172.16.190.218 asm netcool data-asm-elasticsearch-3 75 /opt/ibm/netcool/asm/data/elasticsearch
$ createPersistentVolume 172.16.191.196 asm netcool data-asm-elasticsearch-4 75 /opt/ibm/netcool/asm/data/elasticsearch
$ ssh root@172.16.190.218 mkdir -p /opt/ibm/netcool/asm/data/elasticsearch
$ ssh root@172.16.191.196 mkdir -p /opt/ibm/netcool/asm/data/elasticsearch
```

2. Update deployment configuration. Update the installation configuration with the desired number of Elasticsearch nodes in the cluster. For clarity, only the Elasticsearch cluster size is shown here.

```
global:
  elasticsearch:
    replicaCount: 5
```

3. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

4. Once the upgrade completes, check that the additional Elasticsearch pods are ready.

   Note: The new pods should become ready, and then the existing pods will be updated with the new configuration via a rolling-update, each pod updated in turn while waiting for each to become ready before updating the next. During this process some pods may restart waiting for a quorum of master nodes. The elected master will often change a few times, and this can sometimes happen just prior to a node being updated.

```
$ watch kubectl get pods -l release=asm,app=elasticsearch --namespace=netcool
NAME                  READY   STATUS    RESTARTS   AGE
asm-elasticsearch-0   1/1     Running   0          4m4s
asm-elasticsearch-1   1/1     Running   1          5m52s
asm-elasticsearch-2   1/1     Running   1          7m35s
asm-elasticsearch-3   1/1     Running   1          9m43s
asm-elasticsearch-4   1/1     Running   1          9m43s
```

5. Monitor the state of the Search pods. Search may restart with the new configuration.

```
$ watch kubectl get pod -lapp=search,release=asm
NAME                          READY   STATUS    RESTARTS   AGE
asm-search-5485cf6579-xsdsp   1/1     Running   0          10m
```

6. Verify that the shards are now distributed across all nodes. Check the cluster status according to each node:

```
for pod in `kubectl get pod -l app=elasticsearch | grep -v NAME | awk '{print $1}'`; do
  echo -n "$pod cluster status = "
  kubectl exec $pod -- curl -s localhost:9200/_cluster/health | jq .status
done
```

The system output should indicate a cluster status of 'green'. Yellow would be functioning, although perhaps without the required number of replicas during a node outage, while a status of Red would indicate a problem:

```
asm-elasticsearch-0 cluster status = "green"
asm-elasticsearch-1 cluster status = "green"
asm-elasticsearch-2 cluster status = "green"
asm-elasticsearch-3 cluster status = "green"
asm-elasticsearch-4 cluster status = "green"
```

7. Check the cluster health. View the unfiltered cluster health as in the following example:

```
for pod in `kubectl get pod -l app=elasticsearch | grep -v NAME | awk '{print $1}'`; do
  echo -n "$pod cluster status = "
  kubectl exec $pod -- curl -s localhost:9200/_cluster/health | jq
done
```

Example system output for one node providing full cluster health details:

```
asm-elasticsearch-0 cluster status = {
  "cluster_name": "elastic_production",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 5,
  "number_of_data_nodes": 5,
  "active_primary_shards": 20,
  "active_shards": 60,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

8. Check the node health. View a summary of node health as in the following example:

```
for pod in `kubectl get pod -l app=elasticsearch | grep -v NAME | awk '{print $1}'`; do
  echo "$pod node health"
  kubectl exec $pod -- curl -s localhost:9200/_cat/nodes?v
  echo
done
```

Example system output for two nodes (all nodes should agree who the master node is):

```
asm-elasticsearch-0 node health
ip          heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
10.1.87.147           12        -756   3    0.38    0.36     0.42 mdi       -      l0th8s1
10.1.54.116            8        -675   3    1.37    1.39     0.96 mdi       -      ax1XbEG
10.1.205.83           14        -716   3    0.27    0.37     0.61 mdi       *      1A5OuMp
10.1.91.100           14        -170   5    0.47    0.71     0.75 mdi       -      C4tNw5s
10.1.34.229            7          79  14    2.28    3.30     4.17 mdi       -      TlsN1kz

asm-elasticsearch-1 node health
ip          heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
10.1.205.83           14        -716   3    0.27    0.37     0.61 mdi       *      1A5OuMp
10.1.34.229            7          79  14    2.28    3.30     4.17 mdi       -      TlsN1kz
10.1.91.100           14        -170   5    0.47    0.71     0.75 mdi       -      C4tNw5s
10.1.87.147           12        -756   3    0.38    0.36     0.42 mdi       -      l0th8s1
10.1.54.116            8        -675   3    1.37    1.39     0.96 mdi       -      ax1XbEG
```

9. Check the shard health. View the shard status as in the following example:

```
for pod in `kubectl get pod -l app=elasticsearch | grep -v NAME | awk '{print $1}'`; do
  echo "$pod shard status"
  kubectl exec $pod -- curl -s localhost:9200/_cat/shards?v
  echo
done
```

Example system output for one node (all nodes should report the same status), and for one index (for clarity):

```
asm-elasticsearch-4 shard status
index                       shard prirep state   docs store ip          node
searchservice_v8            1     p      STARTED  102 2.7mb 10.1.54.116 ax1XbEG
searchservice_v8            1     r      STARTED  102 2.7mb 10.1.34.229 TlsN1kz
searchservice_v8            1     r      STARTED  102 2.9mb 10.1.87.147 l0th8s1
searchservice_v8            4     r      STARTED  115 2.9mb 10.1.91.100 C4tNw5s
searchservice_v8            4     r      STARTED  115 2.9mb 10.1.205.83 1A5OuMp
searchservice_v8            4     p      STARTED  115 2.9mb 10.1.87.147 l0th8s1
searchservice_v8            3     r      STARTED  110 2.7mb 10.1.91.100 C4tNw5s
searchservice_v8            3     r      STARTED  110 2.7mb 10.1.54.116 ax1XbEG
searchservice_v8            3     p      STARTED  110 2.7mb 10.1.87.147 l0th8s1
searchservice_v8            2     r      STARTED  102 2.8mb 10.1.91.100 C4tNw5s
searchservice_v8            2     p      STARTED  102 2.8mb 10.1.205.83 1A5OuMp
```

```
searchservice_v8            2     r      STARTED  102 2.8mb 10.1.34.229 TlsN1kz
searchservice_v8            0     r      STARTED   97 2.2mb 10.1.54.116 ax1XbEG
searchservice_v8            0     p      STARTED   97 2.2mb 10.1.205.83 1A5OuMp
searchservice_v8            0     r      STARTED   97 2.2mb 10.1.34.229 TlsN1kz
```

Note the following:

- There are five shards per index.
- Each shard has one primary and two replicas.
- Primary and replica shards are all started (not UNASSIGNED).
- Primary and replica shards are spread over different nodes.
- Primary and replica shards should contain the same number of docs.

Scale in

In the following example the use of five nodes is scaled back to three nodes.

10. Update the deployment configuration to remove one Elasticsearch node: In the following example, the node count (**replicaCount**) in the installation configuration is reduced from five nodes to four:

```
global:
  elasticsearch:
    replicaCount: 4
```

11. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

12. Once the upgrade completes, check the status of the ElasticSearch pods.
    - The number of pods will reduce, and a rolling update will happen to apply the new expected number of nodes.

```
$ watch kubectl get pods -l release=asm,app=elasticsearch --namespace=netcool
NAME                 READY   STATUS    RESTARTS   AGE
asm-elasticsearch-0   1/1     Running   0          2m6s
asm-elasticsearch-1   1/1     Running   0          4m9s
asm-elasticsearch-2   1/1     Running   1          5m24s
asm-elasticsearch-3   1/1     Running   0          6m49s
```

    - During this process some the cluster health will often report as **yellow**, meaning operational, but not with the desired number of replicas. After the scale in, cluster health should return to green:

```
asm-elasticsearch-0 cluster status = "green"
asm-elasticsearch-1 cluster status = "green"
asm-elasticsearch-2 cluster status = "green"
asm-elasticsearch-3 cluster status = "green"
```

13. To further reduce the number of Elasticsearch nodes from four to three, repeat steps 10 to 12.
14. Verify cluster status, node health and shard health by following steps 6 to 9.
15. Deprovision the additional storage (or persistent volumes). See the related “Scaling horizontally: Cassandra database” on page 234 topic for an example of storage deprovisioning.

Note: Always make absolutely sure that you are cleaning the correct nodes.

Related information:

- Elasticsearch horizontal scaling
- Elasticsearch cluster health
- Elasticsearch failover

Scaling horizontally: Kafka message bus

All messages on Kafka are organized into topics. Agile Service Manager has several different topics for different purposes. These topics are further divided into partitions, and these partitions are spread over the available brokers (instance of Kafka in the cluster). Partitions have replicas, but all reads and writes happen on the leader for a partition.

About this task

You can **list the topics in Kafka** as in the following example:

```
$ ./bin/kafka-topics.sh --zookeeper $ZOOKEEPER_URL --list
__consumer_offsets
itsm.monitor.json
itsm.nodes.json
itsm.rebroadcast.json
itsm.resources.json
kafka.topic.notification.json
providers.json
resources.json
status.json
```

You can **describe the topics in Kafka** as in the following example, which shows the number of partitions, how those partitions are spread over the brokers, who is the leader, and where the partition replicas are:

```
$ ./bin/kafka-topics.sh --zookeeper $ZOOKEEPER_URL --describe --topic resources.json
Topic:resources.json PartitionCount:24 ReplicationFactor:3 Configs:
 Topic: resources.json Partition: 0 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
 Topic: resources.json Partition: 1 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
 Topic: resources.json Partition: 2 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
 Topic: resources.json Partition: 3 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
 Topic: resources.json Partition: 4 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
 Topic: resources.json Partition: 5 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
 Topic: resources.json Partition: 6 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
 Topic: resources.json Partition: 7 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
 Topic: resources.json Partition: 8 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
 Topic: resources.json Partition: 9 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
 Topic: resources.json Partition: 10 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
 Topic: resources.json Partition: 11 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
 Topic: resources.json Partition: 12 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
 Topic: resources.json Partition: 13 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
 Topic: resources.json Partition: 14 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
 Topic: resources.json Partition: 15 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
 Topic: resources.json Partition: 16 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
 Topic: resources.json Partition: 17 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
 Topic: resources.json Partition: 18 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
 Topic: resources.json Partition: 19 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
 Topic: resources.json Partition: 20 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
 Topic: resources.json Partition: 21 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
 Topic: resources.json Partition: 22 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
 Topic: resources.json Partition: 23 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
```

Scale out

To scale out Kafka, you provision extra storage, update the deployment configuration, and then perform a helm upgrade.

To redistribute the topic partitions over all available brokers, you then reassign topic partitions.

Scale in

To scale in Kafka, you reassign topic partitions, update the deployment configuration, perform a helm upgrade, and deprovision storage.

Procedure

Scale out

1. Provision extra storage. Before you scale Kafka, you will need to provision extra storage, as also described in more detail in the “Scaling horizontally: Persistence” on page 233 topic. You must provision the additional storage on worker nodes other than the current ones being used, so that the Kafka cluster remains resilient to node failures, as in the following example:

```
$$ source pak_extensions/common/kubhelper.sh
$ createPersistentVolume 172.16.190.218 asm netcool data-asm-kafka-3 15 /opt/ibm/netcool/asm/data/kafka
$ createPersistentVolume 172.16.191.196 asm netcool data-asm-kafka-4 15 /opt/ibm/netcool/asm/data/kafka
$ createPersistentVolume 172.16.192.211 asm netcool data-asm-kafka-5 15 /opt/ibm/netcool/asm/data/kafka
$ ssh root@172.16.190.218 mkdir -p /opt/ibm/netcool/asm/data/kafka
$ ssh root@172.16.191.196 mkdir -p /opt/ibm/netcool/asm/data/kafka
$ ssh root@172.16.192.211 mkdir -p /opt/ibm/netcool/asm/data/kafka
```

2. Update deployment configuration. Update the installation configuration with the desired number of Kafka nodes in the cluster. For clarity, only the Kafka cluster size is shown here.

```
global:
  kafka:
    clusterSize: 6
```

3. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

4. Once the upgrade completes, check that the additional Kafka pods are ready.

```
$ kubectl get pod -lapp=kafka,release=asm
NAME          READY   STATUS    RESTARTS   AGE
asm-kafka-0   2/2     Running   0          14h
asm-kafka-1   2/2     Running   0          14h
asm-kafka-2   2/2     Running   0          14h
asm-kafka-3   2/2     Running   0          12m
asm-kafka-4   2/2     Running   0          12m
asm-kafka-5   2/2     Running   0          12m
```

Reassign topic partitions

Note: In this example, six Kafka brokers are running. If you describe a Kafka topic as described above, it will show that the topics are only distributed over the original three brokers. The topic partitions must be distributed over all available brokers.

5. Define topic to move or reassign.

   Run the following commands inside one of the existing Kafka pods. The following topics are the Agile Service Manager topics:

```
cat <<EOF | tee /tmp/topics-to-move.json
{
  "topics": [
    {
      "topic": "itsm.monitor.json"
    },
    {
      "topic": "itsm.nodes.json"
    },
    {
      "topic": "itsm.rebroadcast.json"
    },
    {
      "topic": "itsm.resources.json"
    },
    {
      "topic": "providers.json"
    },
    {
      "topic": "resources.json"
    },
    {
      "topic": "status.json"
    },
    {
      "topic": "__consumer_offsets"
    },
    {
      "topic": "kafka.topic.notification.json"
    }
  ],
  "version": 1
}
EOF
```

6. Generate a topic reassignment plan. Based on the topic list from the previous step, run the following command to generate a plan:

```
kafka-reassign-partitions.sh --generate
```

   The list of available brokers, six in this example, is displayed by `--broker-list` and can be checked by running the following command:

```
$ /opt/kafka/bin/zookeeper-shell.sh $ZOOKEEPER_URL <<< "ls /brokers/ids"
Connecting to asm-zookeeper:2181 ...
[0, 1, 2, 3, 4, 5]
```

To save the proposed partition reassignment configuration to file:

```
/opt/kafka/bin/kafka-reassign-partitions.sh --generate --zookeeper asm-zookeeper:2181 --topics-to-move-json-file
/tmp/topics-to-move.json --broker-list 0,1,2,3,4,5 | grep version | tail -1 > /tmp/reassignment-plan.json
```

7. Execute the reassignment plan:

```
/opt/kafka/bin/kafka-reassign-partitions.sh --execute --zookeeper $ZOOKEEPER_URL --reassignment-json-file
/tmp/reassignment-plan.json
```

8. Verify the topic reassignment.

   Kafka will redistribute the leaders and replicas amongst all the specified brokers, which may take some time:

```
$ /opt/kafka/bin/kafka-topics.sh --zookeeper $ZOOKEEPER_URL --describe --topic resources.json
Topic:resources.json PartitionCount:24 ReplicationFactor:3 Configs:
 Topic: resources.json Partition: 0 Leader: 1 Replicas: 1,3,4 Isr: 1,3,4
 Topic: resources.json Partition: 1 Leader: 2 Replicas: 2,4,5 Isr: 5,2,4
 Topic: resources.json Partition: 2 Leader: 3 Replicas: 3,5,0 Isr: 0,5,3
 Topic: resources.json Partition: 3 Leader: 4 Replicas: 4,0,1 Isr: 1,0,4
 Topic: resources.json Partition: 4 Leader: 5 Replicas: 5,1,2 Isr: 2,1,5
 Topic: resources.json Partition: 5 Leader: 0 Replicas: 0,2,3 Isr: 0,2,3
 Topic: resources.json Partition: 6 Leader: 1 Replicas: 1,4,5 Isr: 5,1,4
 Topic: resources.json Partition: 7 Leader: 2 Replicas: 2,5,0 Isr: 2,0,5
 Topic: resources.json Partition: 8 Leader: 3 Replicas: 3,0,1 Isr: 0,1,3
 Topic: resources.json Partition: 9 Leader: 4 Replicas: 4,1,2 Isr: 1,2,4
 Topic: resources.json Partition: 10 Leader: 5 Replicas: 5,2,3 Isr: 5,2,3
 Topic: resources.json Partition: 11 Leader: 0 Replicas: 0,3,4 Isr: 0,3,4
 Topic: resources.json Partition: 12 Leader: 1 Replicas: 1,5,0 Isr: 1,5,0
 Topic: resources.json Partition: 13 Leader: 2 Replicas: 2,0,1 Isr: 2,1,0
 Topic: resources.json Partition: 14 Leader: 3 Replicas: 3,1,2 Isr: 2,1,3
 Topic: resources.json Partition: 15 Leader: 4 Replicas: 4,2,3 Isr: 2,3,4
 Topic: resources.json Partition: 16 Leader: 5 Replicas: 5,3,4 Isr: 5,3,4
 Topic: resources.json Partition: 17 Leader: 0 Replicas: 0,4,5 Isr: 0,5,4
 Topic: resources.json Partition: 18 Leader: 1 Replicas: 1,0,2 Isr: 1,2,0
 Topic: resources.json Partition: 19 Leader: 2 Replicas: 2,1,3 Isr: 2,1,3
 Topic: resources.json Partition: 20 Leader: 3 Replicas: 3,2,4 Isr: 2,3,4
 Topic: resources.json Partition: 21 Leader: 4 Replicas: 4,3,5 Isr: 5,3,4
 Topic: resources.json Partition: 22 Leader: 5 Replicas: 5,4,0 Isr: 0,5,4
 Topic: resources.json Partition: 23 Leader: 0 Replicas: 0,5,1 Isr: 0,1,5
```

Scale in

Scaling in is essentially the same operation, where the reassignment plan it to move topics to the brokers that will remain after the scale in. In this example, six brokers are scaled in to three brokers.

9. Reassign topics: Reassign the same topics as for the scale out example as described (from) here. Adjust the reassignment plan to use the brokers that remain after the scale in, in this case:

   ```
   --broker-list    0,1,2
   ```

   Verify the topics are reassigned to the first three brokers, as such:

```
Topic:resources.json PartitionCount:24 ReplicationFactor:3 Configs:
 Topic: resources.json Partition: 0 Leader: 1 Replicas: 1,2,0 Isr: 0,1,2
 Topic: resources.json Partition: 1 Leader: 2 Replicas: 2,0,1 Isr: 0,1,2
 Topic: resources.json Partition: 2 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
 Topic: resources.json Partition: 3 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
 Topic: resources.json Partition: 4 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
 Topic: resources.json Partition: 5 Leader: 0 Replicas: 0,2,1 Isr: 2,0,1
 Topic: resources.json Partition: 6 Leader: 1 Replicas: 1,2,0 Isr: 0,1,2
 Topic: resources.json Partition: 7 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
 Topic: resources.json Partition: 8 Leader: 0 Replicas: 0,1,2 Isr: 1,0,2
 Topic: resources.json Partition: 9 Leader: 1 Replicas: 1,0,2 Isr: 2,1,0
 Topic: resources.json Partition: 10 Leader: 2 Replicas: 2,1,0 Isr: 0,1,2
 Topic: resources.json Partition: 11 Leader: 0 Replicas: 0,2,1 Isr: 0,1,2
 Topic: resources.json Partition: 12 Leader: 1 Replicas: 1,2,0 Isr: 1,0,2
 Topic: resources.json Partition: 13 Leader: 2 Replicas: 2,0,1 Isr: 2,1,0
 Topic: resources.json Partition: 14 Leader: 0 Replicas: 0,1,2 Isr: 2,1,0
 Topic: resources.json Partition: 15 Leader: 1 Replicas: 1,0,2 Isr: 0,1,2
 Topic: resources.json Partition: 16 Leader: 2 Replicas: 2,1,0 Isr: 0,1,2
 Topic: resources.json Partition: 17 Leader: 0 Replicas: 0,2,1 Isr: 0,1,2
 Topic: resources.json Partition: 18 Leader: 1 Replicas: 1,2,0 Isr: 2,1,0
 Topic: resources.json Partition: 19 Leader: 2 Replicas: 2,0,1 Isr: 2,1,0
 Topic: resources.json Partition: 20 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
 Topic: resources.json Partition: 21 Leader: 1 Replicas: 1,0,2 Isr: 0,1,2
 Topic: resources.json Partition: 22 Leader: 2 Replicas: 2,1,0 Isr: 0,1,2
 Topic: resources.json Partition: 23 Leader: 0 Replicas: 0,2,1 Isr: 1,0,2
```

10. Update deployment configuration. Update the installation configuration with the desired number of Kafka nodes in the cluster. For clarity, only the Kafka cluster size is shown here.

```
global:
  kafka:
    clusterSize: 3
```

11. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

12. Deprovision the additional storage (or persistent volumes). See the related "Scaling horizontally: Cassandra database" on page 234 topic for an example of storage deprovisioning.

    Note: Always make absolutely sure that you are cleaning the correct nodes.

Scaling horizontally: Zookeeper synchronization service

Zookeeper synchronization service brokers can be scaled out and scaled in. As Zookeeper requires a majority, it is necessary to use an odd number of pods in the ensemble.

About this task

Note: Agile Service Manager Version 1.1.5 does **not** support scaling for the Zookeeper icomponent.

Note: To avoid an inconsistent lists of servers, you **must** perform scaling via `helm upgrade` rather than the Kubernetes `kubectl scale` command.

Scale out

To scale out Zookeeper, you provision extra storage, update the deployment configuration, and then perform a helm upgrade.

Scale in

To scale in Zookeeper, you update the deployment configuration, perform a helm upgrade, and then deprovision storage.

Procedure

Scale out

1. Provision extra storage. Before you scale out Zookeeper, you must provision extra storage (also described in more detail in the "Scaling horizontally: Persistence" on page 233 topic). You must provision the additional storage on worker nodes other than the current ones being used, so that the Zookeeper ensemble remains resilient to node failures, as in the following example:

```
$ source pak_extensions/common/kubhelper.sh
$ createPersistentVolume 172.16.190.218 asm netcool data-asm-zookeeper-3 15 /opt/ibm/netcool/asm/data/zookeeper
$ createPersistentVolume 172.16.191.196 asm netcool data-asm-zookeeper-4 15 /opt/ibm/netcool/asm/data/zookeeper
$ ssh root@172.16.190.218 mkdir -p /opt/ibm/netcool/asm/data/zookeeper
$ ssh root@172.16.191.196 mkdir -p /opt/ibm/netcool/asm/data/zookeeper
```

2. Update deployment configuration. Update the installation configuration with the desired number of Zookeeper nodes in the cluster. For clarity, only the Zookeeper cluster size is shown here.

```
global:
  zookeeper:
    clusterSize: 5
```

3. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

4. Once the upgrade completes, check that the additional Zookeeper pods are ready.

```
$ watch kubectl get pods -l release=asm,app=zookeeper --namespace=netcool
NAME              READY   STATUS    RESTARTS   AGE
asm-zookeeper-0   1/1     Running   1          17h
asm-zookeeper-1   1/1     Running   0          17h
asm-zookeeper-2   1/1     Running   1          17h
asm-zookeeper-3   1/1     Running   0          3m31s
asm-zookeeper-4   1/1     Running   0          3m31s
```

5. Also monitor the state of the Kafka pods.

```
$ watch kubectl get pod -lapp=kafka,release=asm
NAME          READY   STATUS    RESTARTS   AGE
asm-kafka-0   2/2     Running   0          30m
asm-kafka-1   2/2     Running   0          32m
asm-kafka-2   2/2     Running   0          33m
asm-kafka-3   2/2     Running   0          35m
asm-kafka-4   2/2     Running   0          36m
asm-kafka-5   2/2     Running   0          38m
```

6. Check that one of the Zookeeper nodes has been elected as leader:

```
for pod in `kubectl get pod -l app=zookeeper,release=asm | grep -v NAME | awk '{print $1}'`; do
  echo -n "$pod ";
  echo "stat" | kubectl exec -i $pod -- socat - tcp:localhost:2181 | grep Mode;
done
```

Example output

```
asm-zookeeper-0 Mode: follower
asm-zookeeper-1 Mode: leader
asm-zookeeper-2 Mode: follower
asm-zookeeper-3 Mode: follower
asm-zookeeper-4 Mode: follower
```

7. To see the full Zookeeper node statistics, run the following command:

```
for pod in `kubectl get pod -l app=zookeeper,release=asm | grep -v NAME | awk '{print $1}'`; do
  echo "$pod stat:";
  echo "stat" | kubectl exec -i $pod -- socat - tcp:localhost:2181;
  echo;
done
```

Scale in

Scaling in is almost the same operation as scaling out, but in reverse.

8. Update deployment configuration. Update the installation config with the desired number of Zookeeper nodes in the ensemble. For clarity, only the Zookeeper ensemble size is shown here.

```
global:
  zookeeper:
    clusterSize: 3
```

9. Perform a helm upgrade using the updated configuration:

```
helm upgrade asm icp-local/ibm-netcool-asm-prod --values=asm-config.yaml --tls
```

10. Deprovision the additional storage (or persistent volumes). For an example, see the equivalent storage deprovisioning step in the related Cassandra database scaling topic.

Note: Always make absolutely sure that you are cleaning the correct nodes.

Related information:

- Zookeeper clustered (multi-server) setup
- Zookeeper common problems

System health and logging

Docker containers have built-in health monitoring, which you can use to check if a service is still available. In addition, you can use configurable logging functionality to monitor system health and assist in troubleshooting.

Performing a health check from the UI using Docker Observer

You can access your Netcool Agile Service Manager deployment's system health information as reported by the Docker Observer in the Topology Viewer.

Additional actions > View System Health
Open the **Additional actions** (**...**) drop-down menu, and then use the **View System Health** option to access your Netcool Agile Service Manager deployment's system health information.

Tip: For more information on using the Docker Observer, see the following topic: "Defining Docker Observer jobs" on page 118

Configuring logging for the Netcool Agile Service Manager UI

The log file directory contains two sets of files, standard log files and trace files. The standard logs store high level log messages, and the trace files store low level log messages. You can configure the logging levels, log file location, as well as file count and size for the Netcool Agile Service Manager UI logs.

About this task

The Netcool Agile Service Manager UI server routinely records information about its operations in log files. By default the trace level is set relatively high to prevent logs from being written to disk during the normal, healthy running of the system. However, if a problem occurs in the system which requires investigation, the trace level may be manually, and temporarily, changed to a lower, more detailed level.

Tip: Use trace level logging sparingly, as it may adversely affect performance.

The default log file location is *DASH_PROFILE*/logs/inasm

Example log file location
/opt/IBM/JazzSM/profile/logs/inasm

Most logging for the Netcool Agile Service Manager UI occurs during system startup, that is, when the DASH server is starting, and also when data requests are made from the topology viewer to the topology service.

You can configure the following Netcool Agile Service Manager logging parameters to suit your specific requirements:

- Log file location
- Logging levels
- Log file names
- Maximum log file size
- Maximum log file count

The log configuration settings are stored in the Netcool Agile Service Manager UI application configuration file, which is located at $NCHOME/inasm/etc/application.yml

Example application config file location

```
/opt/IBM/netcool/gui/inasm/etc/application.yml
```

Note: The initialization of the logging system is one of the startup operations for the Netcool Agile Service Manager UI. Until this initialization has completed, it is not possible to send messages to the log or trace files. During this initialization phase, any important log messages will instead be sent to the DASH system log file, which is located at *DASH_PROFILE*/logs/server1/SystemOut.log

Example location

```
/opt/IBM/JazzSM/profile/logs/server1/SystemOut.log
```

Procedure

Edit the application configuration file

1. Open the application config file using an appropriate editor.
2. Edit the following settings:

logDirectory

The directory in which the ASM log and trace files should be stored, relative to the JazzSM profile directory.

The default value is `/logs/inasm`

logLevel

The lowest level of messages to report in the log files.

Valid options, from lowest to highest level, are:

- ALL
- FINEST
- FINER
- PROFILE
- FINE
- CONFIG
- INFO
- AUDIT
- WARNING
- SEVERE
- OFF

Remember: Use trace level logging sparingly, as it may adversely affect performance. The more fine-grained the level of logging, the more detail will be written to the logs, thereby affecting the performance.

The default value is `INFO`

logFilename

The filename pattern for log files.

`%g` may be used to represent the backup log file number.

The default value is `inasm.%g.log`

logCount

The number of backup log files to keep.

The default value is `5`

traceLevel
The lowest level of messages to report in the trace files.

Valid options, from lowest to highest level, are:

- ALL
- FINEST
- FINER
- PROFILE
- FINE
- CONFIG
- INFO
- AUDIT
- WARNING
- SEVERE
- OFF

The default value is `CONFIG`

traceFilename
The filename pattern for trace files. `%g` may be used to represent the backup trace file number.

The default value is `inasm.%g.trace`

traceMaxSize
The maximum size that trace files should be allowed to grow to, in MB. Once a trace file reaches the maximum size, the file is renamed and kept as a backup.

The default value is `10`

traceCount
The number of backup trace files to keep.

The default value is `5`

Restart DASH to allow the changes to take effect.

3. To stop the DASH server, run `<DASH_PROFILE>/bin/stopServer.sh server1`
4. Once stopped, start the DASH server: `<DASH_PROFILE>/bin/startServer.sh server1`

Viewing the service logs (on-prem)

Logs for all Netcool Agile Service Manager services can be found in the `$ASM_HOME/logs/<service>` directories. You can set logging levels for the user-facing services, such as the observers and search, using scripts provided.

About this task

Table 51. Log names and directories for Netcool Agile Service Manager services

Service	Directory	Log
Event Observer	`$ASM_HOME/logs/event-observer`	event-observer.log
ITNM Observer	`$ASM_HOME/logs/itnm-observer`	itnm-observer.log

Table 51. Log names and directories for Netcool Agile Service Manager services (continued)

Service	Directory	Log
OpenStack Observer	`$ASM_HOME/logs/openstack-observer`	openstack-observer.log
File Observer	`$ASM_HOME/logs/file-observer`	file-observer.log
Docker Observer	`$ASM_HOME/logs/docker-observer`	docker-observer.log
Topology service	`$ASM_HOME/logs/topology`	topology-service.log
Search service	`$ASM_HOME/logs/search`	search-service.log
Elasticsearch	`$ASM_HOME/logs/elasticsearch`	elasticsearch.log
Cassandra database	`$ASM_HOME/logs/cassandra`	system.log
Kafka message bus	`$ASM_HOME/logs/kafka`	server.log
Zookeeper synchronization service	`$ASM_HOME/logs/zookeeper`	zookeeper.log
UI API service (see UI API service logging)	`$ASM_HOME/logs/ui-api`	ui-api.log

When a log reaches its maximum size, it is compressed, date-stamped and versioned, and moved to a subdirectory, for example:

```
$ASM_HOME/logs/topology/tmp/topology-service-2017-03-23-0.log.gz
```

and then

```
$ASM_HOME/logs/topology/tmp/topology-service-2017-03-23-1.log.gz
```

Table 52. Scripts to configure the logging levels for Netcool Agile Service Manager services

Service	Log level script
Event Observer	event_observer_log_level.sh
ITNM Observer	itnm_observer_log_level.sh
OpenStack Observer	openstack_observer_log_level.sh
File Observer	file_observer_log_level.sh
Docker Observer	docker_observer_log_level.sh
Topology Service	topology_service_log_level.sh
Search Service	search_service_log_level.sh

UI API service logging: The Agile Service Manager UI uses the UI API to retrieve data from the other Agile Service Manager services. Data retrieval errors are recorded in the UI API log.
At a log level of CONFIG or above, which is the default log level, the UI API log only contains information about the service startup, as well as any high level warnings and errors. To diagnose a problem, a lower log level may be useful. For example, a FINE level logs all incoming requests and an indication of success, whereas at FINER and FINEST additional details are logged about how each REST request is handled.
The UI API service does not have a shell script, which means you must configure

logging for the UI API manually using Curl commands.
You can view your current log levels, or change them:

See current log level

To view your UI API service log level, change the following parameters in the example cURL command to reflect your own deployment:

- Nginx host and port
- Username (if other than default)
- Password (if other than default)

```
curl -X GET -H  "accept: application/json" -u asm:asm -k
https://asm-nginx-host:443/1.0/ui-api/service/config
```

Change current log level

Again, for your own deployment you must change the Nginx host and port, as well as the username and password (if other than default).

The following values are allowed when setting the log level for the UI API:

- ALL
- FINEST
- FINER
- FINE
- CONFIG (default)
- INFO
- AUDIT
- WARNING
- SEVERE
- OFF

```
curl -X POST -H "Content-Type: application/json" -u asm:asm -k -d "{\"logLevel\":
\"FINER\"}" https://asm-nginx-host:443/1.0/ui-api/service/config
```

Viewing the service logs (ICP)

You can see the logs for all Netcool Agile Service Manager ICP services using the `kubectl logs` command.

About this task

All Netcool Agile Service Manager ICP containers are deployed in pods. You can identify logs either by their names or their labels.

Procedure

1. To list all Agile Service Manager ICP pods and the labels applied to the pods, run the `kubectl get pod --show-labels` command, as in the example below.
2. Display the logs using either the pod names or labels. When running the `kubectl logs` command, you must also specify the namespace.

   View logs for a specific pod

   ```
   kubectl logs asm-layout-88bd88bdb-htjfp --namespace=netcool
   ```

   View logs using the label

   ```
   kubectl logs -l app=layout --namespace=netcool
   ```

Example

When using the `kubectl get pod --show-labels` command to list all pods, the system will display the pod names, their status, number of restarts, age, and labels.

The labels retrieved will also contain additional information, such as, for example, the release name, which is important if more than one Agile Service Manager deployment exists.

```
$ kubectl get pod --show-labels
NAME                                          READY     STATUS    RESTARTS   AGE
LABELS
deploy-test-cassandra-0                       1/1       Running   0          2d
app=cassandra,chart=cassandra,controller-revision-hash=deploy-test-cassandra-7d8f56b884,
heritage=Tiller,release=deploy-test,statefulset.kubernetes.io/pod-name=deploy-test-cassandra-0
deploy-test-ciscoaci-observer-8755fcc94-7cp7h      1/1       Running   0          2d
app=ciscoaci-observer,chart=ciscoaci-observer,heritage=Tiller,
pod-template-hash=431197750,release=deploy-test
deploy-test-contrail-observer-54f49cdc8c-bjdnq     1/1       Running   0          2d
app=contrail-observer,chart=contrail-observer,heritage=Tiller,
pod-template-hash=1090578747,release=deploy-test
deploy-test-dns-observer-7ffc598847-kdqlr          1/1       Running   0          2d
app=dns-observer,chart=dns-observer,heritage=Tiller,pod-template-hash=3997154403,
release=deploy-test
deploy-test-elasticsearch-0                        1/1       Running   0          2d
app=elasticsearch,chart=elasticsearch,controller-revision-hash=deploy-test-elasticsearch-5bb4857dff,
heritage=Tiller,release=deploy-test,statefulset.kubernetes.io/pod-name=deploy-test-elasticsearch-0
deploy-test-event-observer-5c85dfb557-79n9w        1/1       Running   0          2d
app=event-observer,chart=event-observer,heritage=Tiller,pod-template-hash=1741896113,
release=deploy-test
deploy-test-file-observer-76c5869d8b-tm7xj         1/1       Running   0          2d
app=file-observer,chart=file-observer,heritage=Tiller,pod-template-hash=3271425846,
release=deploy-test
deploy-test-ibmcloud-observer-65497f5478-4nsr6     1/1       Running   0          2d
app=ibmcloud-observer,chart=ibmcloud-observer,heritage=Tiller,pod-template-hash=2105391034,
release=deploy-test
deploy-test-itnm-observer-85d7cc7878-2xdhf         1/1       Running   0          2d
app=itnm-observer,chart=itnm-observer,heritage=Tiller,pod-template-hash=4183773434,
release=deploy-test
deploy-test-kafka-0                                2/2       Running   0          2d
app=kafka,chart=kafka,controller-revision-hash=deploy-test-kafka-5c78c96dbc,
heritage=Tiller,release=deploy-test,statefulset.kubernetes.io/pod-name=deploy-test-kafka-0
deploy-test-kubernetes-observer-66cb697d7-ktgqt    1/1       Running   0          2d
app=kubernetes-observer,chart=kubernetes-observer,heritage=Tiller,
pod-template-hash=227625383,release=deploy-test
deploy-test-layout-66656685bd-wkp6b                1/1       Running   0          2d
app=layout,chart=layout,heritage=Tiller,pod-template-hash=2221224168,
release=deploy-test
deploy-test-merge-688468b7-q4ckq                   1/1       Running   0          2d
app=merge,chart=merge,heritage=Tiller,pod-template-hash=24402463,
release=deploy-test
deploy-test-newrelic-observer-7c879f5545-72k4p     1/1       Running   0          2d
app=newrelic-observer,chart=newrelic-observer,heritage=Tiller,
pod-template-hash=3743591101,release=deploy-test
deploy-test-openstack-observer-5df8fffd56-kfdjc    1/1       Running   0          2d
app=openstack-observer,chart=openstack-observer,heritage=Tiller,
pod-template-hash=1894999812,release=deploy-test
deploy-test-rest-observer-789cd8699d-zb8vx         1/1       Running   0          2d
app=rest-observer,chart=rest-observer,heritage=Tiller,
pod-template-hash=3457842558,release=deploy-test
deploy-test-search-5dc4ccc99b-nmqz2                1/1       Running   0          2d
app=search,chart=search,heritage=Tiller,pod-template-hash=1870777556,
release=deploy-test
deploy-test-servicenow-observer-854bbff7dc-sl75x   1/1       Running   0          2d
app=servicenow-observer,chart=servicenow-observer,heritage=Tiller,
pod-template-hash=4106699387,release=deploy-test
deploy-test-taddm-observer-79dd5b556-bnzp2         1/1       Running   0          2d
app=taddm-observer,chart=taddm-observer,heritage=Tiller,
pod-template-hash=358816112,release=deploy-test
deploy-test-topology-75688cfc48-jjkfd              1/1       Running   0          2d
app=topology,chart=topology,heritage=Tiller,pod-template-hash=3124479704,
release=deploy-test
deploy-test-vmvcenter-observer-675bd88f5c-8kgnk    1/1       Running   0          2d
app=vmvcenter-observer,chart=vmvcenter-observer,heritage=Tiller,
pod-template-hash=2316844917,release=deploy-test
deploy-test-vmwarensx-observer-dc96946f4-7jcxj     1/1       Running   0          2d
app=vmwarensx-observer,chart=vmwarensx-observer,heritage=Tiller,
pod-template-hash=875250290,release=deploy-test
deploy-test-zookeeper-0                            1/1       Running   0          2d
app=zookeeper,chart=zookeeper,controller-revision-hash=deploy-test-zookeeper-d65f46875,
heritage=Tiller,release=deploy-test,statefulset.kubernetes.io/pod-name=deploy-test-zookeeper-0
```

Chapter 7. Troubleshooting

Use the following topics to troubleshoot specific issues.

Installation troubleshooting

See the following information to troubleshoot installation issues. For ICP-specific installation, upgrade or uninstall issues, see "ICP troubleshooting" on page 254

License issues

During a first installation of Netcool Agile Service manager, or when the license terms change, you will be prompted to accept the software license. If you do not complete this step, the software will not start and this error will occur:

```
ERROR: Couldn't find env file: /opt/ibm/netcool/asm/licenses/
.accept_license
```

Workaround

To review and accept the license terms, use the following command:

```
/opt/ibm/netcool/asm/bin/license-review.sh
```

ASM_HOME variable warnings and errors

When running any docker-compose commands, such as starting or stopping Agile Service Manager, the docker-compose service needs to load the `.env` and `docker-compose.yml` files located in ASM_HOME. If you do not run the docker-compose command from the ASM_HOME directory, warnings and errors like the following may occur:

```
WARNING: The ASM_HOME variable is not set. Defaulting to a blank string.

ERROR: .IOError: [Errno 2] No such file or directory: u'/etc/nasm-docker-
observer.yml'

ERROR: Can't find a suitable configuration file in this directory or any
parent. Are you in the right directory? Supported filenames:
docker-compose.yml, docker-compose.yaml
```

Workaround

You must run docker-compose commands from the ASM_HOME directory.

To ensure that your present working directory is ASM_HOME, you can take the following steps:

Change you current directory

To change your current directory to the ASM_HOME directory, use the **cd** command, as in the following example:

```
$ cd /opt/ibm/netcool/asm
```

Check your current directory

To verify that your current directory is the ASM_HOME directory, use the **pwd** command, as in the following example:

```
$ pwd
/opt/ibm/netcool/asm
```

Startup troubleshooting

See the following information to troubleshoot issues occurring when launching the application.

Cassandra database startup issue

During startup, the topology service attempts to connect to the Cassandra datastore before it has fully started, thereby causing an error message such as the following:

```
ERROR [14:11:07.330] [main] c.i.i.t.g.ConnectionManager - Unexpected
Throwable caught creating TitanGraphjava.lang.IllegalArgumentException:
Could not instantiate implementation:
com.thinkaurelius.titan.diskstorage.cassandra.astyanax.AstyanaxStoreManager
```

Permissions of data and logs directories can result in Agile Service Manager not coming up cleanly with some services continually restarting, e.g. if you delete the directories and then restart without the right environment.

Workaround

None required.

The topology service will try to connect to the Cassandra datastore again, and will succeed once Cassandra is up and running.

Search troubleshooting

See the following information to troubleshoot Search service issues.

Re-indexing Search

If data in Elasticsearch is out of sync with data in the Cassandra database, resynchronize it by calling the `rebroadcast` API of the topology service. This triggers the rebroadcast of all known resources on Kafka, and the Search service will then index those resources in Elasticsearch.

Workaround

Call the `rebroadcast` API of the Topology service, specifying a tenantId:

`https://`*`master_fqdn`*`/1.0/topology/swagger#!/Crawlers/rebroadcastTopology`

Elasticsearch versioning errors

While using the Search service, versioning errors are reported.

Agile Service Manager uses Elasticsearch version 6.5.3, and both search components have been updated accordingly (nasm-elasticsearch, nasm-search).

Workaround

Ensure that you have deployed the latest versions of nasm-elasticsearch and nasm-search.

Elasticsearch running out of disk space

If Elasticsearch runs out of disk space (when the disk is 95% full or more), it places index blocks into read-only mode, which can result in the following error: `ClusterBlockException[blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];]`

This issue manifests itself as the inability to search for new or potentially updated resources that are present in the topology service because they have not been indexed by the search service. It is recommended to proactively monitor disk space and take preventative action as necessary should disk space become low.

Workaround

Make more space available on the disk, and then manually release the locked index by issuing the following curl command:

```
curl --user <asm_user>:<asm_password> -k -XPUT "https://<asm_host>/
elasticsearch/searchservice_v8/_settings" -H 'Content-Type: application/json' -d'
{
"index.blocks.read_only_allow_delete": null
}
'
```

Restart Elasticsearch.

Search returns data for up to 12 hours longer than it exists in the topology service

Restriction: Agile Service Manager interprets the timeToLive settings in the Topology service and Search service differently. While the topology service can remove deleted data at one minute intervals, the Search service removes data every 12 hours only. Therefore if both services have the same timeToLive settings, for example 48 hours, then the Search service will continue to 'find' that data for up to an additional 12 hours after it has been removed from the topology service.

Workaround

If you want to ensure that Search can never return data that the topology service has already deleted, ensure that the number of timeToLive days in the Search service is one fewer than the number of days set in the topology service.

Remember: As a result of this workaround the Search service will potentially be unable to return some data that exists in the topology service for a period of up to 12 hours.

Observer troubleshooting

See the following information to troubleshoot a variety of observer issues.

File Observer heap size issue

If a large number of events are being processed, the default Java Virtual Machine (JVM) memory settings may prove insufficient and processing errors may occur. These errors can generate WARNING logs, and processing of data may be suspended.

Workaround

Increase the maximum Java heap size (Xmx) value to 6G.

1. Edit the `ASM_HOME/etc/nasm-file-observer.yml` file and change the Xmx value in the following default argument to 6G:

   ```
   JVM_ARGS: ${FILE_OBSERVER_JVM_ARGS:--Xms1G -Xmx2G}
   ```

2. Restart the service.

Other troubleshooting

See the following information to troubleshoot a variety of service issues, such proxy server buffering warning.

Proxy service buffering warning

If large information payloads are sent to the Nginx proxy server service, the error log may record the following warning: `[warn]...a client request body is buffered to a temporary file...`

Such warnings indicate that Nginx is temporarily storing the payload in storage as opposed to using memory. While this does not affect the performance of Agile Service Manager much, these messages could flood the log file, making other debugging tasks more difficult.

Workaround

To increase the limit at which Nginx uses memory rather than storage, open the `$ASM_HOME/etc/nginx/conf.d/general.conf` configuration file with a suitable text editor, and increase the value of the **`client_body_buffer_size`** parameter as required.

Restart the proxy service using the following command:

```
$ASM_HOME/bin/docker-compose restart proxy
```

ICP troubleshooting

See the following information to troubleshoot IBM Cloud Private issues.

Services not binding to storage (after upgrade or uninstall)

Some services fail to bind to the provisioned storage, typically resulting in pods stuck in 'pending' state.

After removing a previous installation of Agile Service Manager and some of its PersistentVolumeClaim (PVC) objects, any associated PersistentVolume (PV) objects are placed in a 'Released' state. They are now unavailable for bonding, even if new PVCs that are part of a new Agile Service Manager installation have the same name and namespace. This is an important security feature to safeguard the previous PV data.

Investigating the problem: The following example lists the 'elasticsearch' pods and their status, and the result shows the 'pending' status, indicating the problem.

```
$ kubectl get pod -l app=elasticsearch
```

```
NAME                    READY  STATUS            RESTARTS  AGE
asm-elasticsearch-0     0/1    ContainerCreating  0        4s
asm-elasticsearch-1     0/1    Pending            0        3s
asm-elasticsearch-2     0/1    Pending            0        3s
```

This example examines the state of the PersistentVolumeClaims and the (truncated) result indicates that the status is 'pending'.

```
$ kubectl get pvc -l app=elasticsearch
```

```
NAME                      STATUS    VOLUME
data-asm-elasticsearch-0  Bound     asm-data-elasticsearch-0
data-asm-elasticsearch-1  Pending
data-asm-elasticsearch-2  Pending
```

This example examines the PersistentVolumes and the (truncated) result indicates that the status is 'released'.

```
$ kubectl get pv -l app=elasticsearch
```

```
NAME                      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
asm-data-elasticsearch-0   75Gi       RWO            Retain           Bound
asm-data-elasticsearch-1   75Gi       RWO            Retain           Released
asm-data-elasticsearch-2   75Gi       RWO            Retain           Released
```

Solution: As admin user, remove the `PV.Spec.ClaimRef.UID` field from the PV objects to make the PV available again. The following (truncated) example shows a PV that is bound to a specific PVC:

```
apiVersion: v1
kind: PersistentVolume
spec:
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: data-asm-elasticsearch-1
    namespace: default
    resourceVersion: "81033"
    uid: 3dc73022-bb1d-11e8-997a-00000a330243
```

To solve the problem, you edit the PV object and remove the `uid` field, after which the PV status changes to 'Available', as shown in the following example:

```
$ kubectl get pv -l app=elasticsearch
```

```
NAME                      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
asm-data-elasticsearch-0   75Gi       RWO            Retain           Bound
asm-data-elasticsearch-1   75Gi       RWO            Retain           Available
asm-data-elasticsearch-2   75Gi       RWO            Retain           Available
```

User interface timeout errors

To prevent or mitigate UI timeout errors on ICP, you can increase the timeout values for the following parameters, which are defined in `configmap`:

- topologyServiceTimeout
- searchServiceTimeout
- layoutServiceTimeout

To change the timeout values of these (in seconds) edit the configmap using the following command:

```
kubectl edit configmap {{ .Release.Name }}-asm-ui-config
```

When done, restart the NOI webgui pod.

The default serviceaccount does not have the required permissions to query the endpoints API

If the initContainers are failing with this error, then you do not have the correct permission to query the endpoints API. Most initContainers in Agile Service Manager query the Kubernetes endpoints API to check and wait for dependent services to be ready.

The required roles are usually created when the Agile Service Manager Helm Chart is installed. If the user deploying the chart does not have permissions to create roles and rolebindings, this error occurs.

Workaround

Ask the cluster administrator to grant you the required role, using the following command:

```
kubectl create clusterrolebinding asm-view-binding
--clusterrole=view --serviceaccount=netcool:default
```

Where:

asm-view-binding
: Is the name of the role binding to create

view
: Is the clusterrole required, which allows read-only access to see most objects in a namespace

netcool
: Is the namespace into which you are installing Agile Service Manager

default
: Is the name of the serviceaccount

Chapter 8. Reference

Use the following reference information to enhance your understanding of Netcool Agile Service Manager interfaces and functionality.

Topology service reference

Use this introduction to the Netcool Agile Service Manager services to understand the most important topology service concepts and functions.

You can access the Swagger documentation for the topology service at the following location: `https://<your host>/1.0/topology/swagger`

Remember: IBM Netcool Agile Service Manager is cloud-born, and built on secure, robust and proven technologies. It is designed to be flexible and can be extended as needed using plug-in components and micro-services to cater for highly specific environments.

Refresher:

It is important that you are familiar with the following important terms introduced in the Glossary, as they will be used and expanded on in this reference section:

resource
A resource is a node in an interconnected topology, sometimes also referred to as a vertex, or simply a node. It can be anything in a user-specific topology that has been designated as such, for example a hardware or virtual device, a location, a user, or an application.

edge An edge is a relationship between resources, also simply referred to as the 'link' between resources.

Edges have a *label*, which allocates them to a family of edges with specific behavior and governs how they are displayed in the UI, and an *edgeType*, which defines the relationship in real terms.

tenant A tenant is represented by a globally unique identifier, its tenant ID.

The default tenant ID is: `cfd95b7e-3bc7-4006-a4a8-a73a79c71255`

provider
A provider is usually a single data source within the scope of a tenant.

Note: A provider's **`uniqueId`** property for a resource is unique only within the scope of a provider.

status Status is a property of one or more resources, and a single resource can have different types of status.

Each status can be in one of three states: open, clear or closed.

The status of a resource can be derived from events, in the case of the resource having been retrieved via the Event Observer, or it can be supplied when resources are posted to the topology service.

Additional: For more information on how topologies are displayed in the UI, you can take a look at the topology viewer screen reference topic:"Topology viewer

reference" on page 293

Properties

The Topology Service has two categories of properties, generic and user. Generic properties have fixed data types, while user-defined properties do not.

Generic properties

Generic properties are few in number and constrained to a fixed data type. They can also be subdivided into those which are read-write and those which are read-only.

uniqueId
The uniqueId is the string used to match resources from the same provider. It could be, for example, a UUID via which the provider can look up its own local data store for information about that device.

If you send the same resource with the same Id and the same provider more than once, it will be treated as the same resource. However, the uniqueId is only unique within the context of its provider.

matchTokens
These tokens are used to store strings which are significant with respect to that resource, and could match it to events.

name The name string is **required** by the UI to display a resource.

This string does not have to be unique, and it should be short and memorable.

tags Tags can be used to filter resources and store strings, which can later retrieve groups of related resources.

entityTypes
These are defined as a set, though with usually only a single member, of the type(s) this resource represents.

Tip: A set of predefined entityTypes with associated icons exist, and it is recommended, though not required, that you use these. See the "Entity types" on page 265 topic for more information.

Table 53. Generic properties (either read-only or read/write)

Name	Type	Cardinality	Alias	Read-only
age	integer	single		no
aliasIds	Id	set	_aliasIds	yes
beginTime	long	single	_startedAt	yes
changeTime	long	single	_modifiedAt	yes
createTime	long	single	_createdAt	yes
deleteTime	long	single		yes
description	string	single		no
edgeTenantId	Id	single	_edgeTenantId	yes
edgeType	string	single	_edgeType	yes
endTime	long	single	_deletedAt	yes
entityTypes	string	set		no
eventId	string	single		yes

Table 53. Generic properties (either read-only or read/write) (continued)

Name	Type	Cardinality	Alias	Read-only
eventManager	string	single		yes
expireTime	long	single	_expiredAt	yes
geolocation	GeoLocation	single		no
hasState	string	single		yes
icon	string	single		no
id	long	single		yes
keyIndexName	string	single		yes
label	string	single		yes
matchTokens	string	set		no
name	string	single		no
observedTime	long	single	_observedAt	yes
operation	string	single		yes
partOfExternal	Boolean	single		yes
prevBeginTime	long	single		yes
providerId	Id	single		yes
providerName	string	single		yes
reconciliation Tokens	string	set		yes
referenceId	Id	single		yes
referenceNo	long	single		yes
serialized HashMap	HashMap	single		yes
severity	string	single		no
speed	long	single		no
statusType	string	single		yes
tags	string	set		no
tenantIds	Id	set	_tenantIds	yes
uniqueId	string	single		no
uuid	Id	single	_id	yes
version	string	single		no
vertexType	string	single		yes

User properties

User-defined properties are free-form, and are not constrained by any given data type. You can add new user properties as needed.

You can define any custom properties, such as, for example **`ipAddress`**.

Note: All user-defined properties such as ipAddress are not in the generic set, and are stored as a serialized 'blob' data type instead. The implication of this storage convention is that these properties cannot be filtered, as they are incompatible with the **`_filter`** query parameter used in the REST API.

Tip: The Swagger documentation listing all properties can be found at the following default location: `http://<your host>:8080/1.0/topology/swagger#!/Schema/getDefaultProperties`

Edge labels

The topology service defines a family of labels for the edges it supports.

Note: Most interactions with edges in the Topology Service are with edge *types* rather than edge *labels*. Edge types can be conceptualized as instances of the edge label classes, and are documented separately here.

aggregation

A 'parent-child' relationship where the parent (source) is aggregating the children (target).

The type of aggregation is determined by the value of the edge type.

Use this edge label to represent a resource in the UI that is composed of various elements, for example a book `contains` words.

- The direction is always from parent to child.
- Children can have multiple parents.

See Table 54 on page 261 for information on the edge types associated with this edge label.

association

A 'weak' relationship where both source and target vertex can exist independently, and neither source nor target are required for the other to function, despite them being related.

The specific type of association is determined by the edge type.

Use this edge label to represent a general relationship between vertices in the UI, for example a person `has` a house.

The label itself has no direction, but a direction could be implied by the edge **type** used.

See Table 55 on page 261 for information on the edge types associated with this edge label.

dataFlow

A dataFlow label represents a data flow between a pair of resources.

The specific type of data flow is qualified by properties on the edge type.

Use this label when you need to represent any form of data flow in the UI, for example a person emailing another person.

- The label itself has no direction, but a direction can be implied from the edge type used.

See Table 56 on page 263 for information on the edge types associated with this edge label.

dependency

A 'strong' relationship where the source depends on the target and cannot operate independently of it.

The specific type of dependency is determined by the edgeType.

Use this label when you need to represent the dependency of one resource on another in the UI, for example a application **dependsOn** a database. The direction is always from the dependent resource to the independent resource.

See Table 57 on page 263 for information on the edge types associated with this edge label.

metaData

A relationship that associates a resource to meta-data 'outside' its actual logical or physical model.

Note: This is meta data, and not displayed in the UI.

Also a relationship between different instances of meta-data.

Use when you need to associate meta-data with a resource to support application behavior, for example a PoP `has` a test definition.

See Table 58 on page 264 for information on the edge types associated with this edge label.

Edge types

All edges created in the Topology Service should have an edge **type** defined. The following section lists the edge types that are associated with each of the public-facing edge labels. If none of the default edge types suffice, you can create custom edge types.

Edge types for public edge labels

Remember: Edge **types** can be thought of as being instances of the edge **label** classes, in this case the public-facing edge labels. Most interactions with edges in the Topology Service are with edge *types* rather than edge *labels*.

You can access the Swagger documentation for 'edge types' at the following default link: https://localhost/1.0/topology/swagger#!/Schema/getEdgeTypes

Table 54. Edge types for the **Aggregation** *edge labels*

Edge type	Description	Example
contains	The source resource is considered to contain the target resource	Slot `contains` a card
federates	The source resource is a federation of the target resources	Database `federates` nodes
members	The source resource has the target resources as its members	Subnet `members` are IP addresses

Table 55. Edge types for the **Association** *edge labels*

Edge type	Description	Example
aliasOf	Depicts that one resource is an alias of another; potentially from different providers	FQDN1 is an `alias` of FQDN2

Table 55. Edge types for the **Association** *edge labels (continued)*

Edge type	Description	Example
assignedTo	When one resource has been assigned to another	The alarm is `assingedTo` an operator
attachedTo	When one resource is attached to another	The plug is `attachedTo` to the cable
classifies	When one resource is used to classify another	The government `classifies` the document
configures	When one resource configures or provides configuration for another resource	The file `configures` the application
deployedTo	Describes when one resource has been deployed to another	The application was `deployedTo` the server
exposes	When one resource exposes another	The application `exposes` the interface
has	Generalized relationship when one resource possesses, owns or has another resource	Host `has` component
implements	When one resource implements another.	The class `implements` the interface
locatedAt	When one resource is physically located in/at another resource's location	Host is `locatedAt` data centre
manages	When one resource manages another	The boss `manages` the employee
monitors	When one resource monitors another	The application `monitors` the host
movedTo	When one resource has moved to a new and different resource	The service has `movedTo` the host
origin	Indicates the origin of a Vertex	Device's `origin` is a vendor
owns	Indicates ownership of one resource by another resource	The user `owns` the server
rates	Can be used when one resource rates another	The manager `rates` the employee
resolvesTo	When one resource resolves to another	The hostname `resolvesTo` an address
realizes	When one resource realizes another	The hypervisor `realizes` the virtual machine
segregates	When one resource segregates another	The firewall `segregates` the network
uses	When one resource takes, employs or deploys another resource as a means of achieving something	Application `uses` this disk

Table 56. Edge types for the **Data flow** *edge labels*

Edge type	Description	Example
accessedVia	One resource is accessed via another, typically remote.	Server is `accessedVia` a FQDN
bindsTo	A network layering relationship such that the source 'runs on top' of the target.	Logical interface `bindsTo` a physical port
communicatesWith	A relationship whereby one resource communicates with another	The sensor `communicatesWith` an application
connectedTo	A relationship whereby one resource is connected to another	The laptop is `connectedTo` the switch
downlinkTo	One resource is down-linked to another	The controller has a `downlinkTo` the sensor
reachableVia	A resource is reachable via another resource	The network is `reachableVia` the gateway
receives	A resource receives data from another	The Mail server `receives` an email
routes	A relationship whereby one resource routes data for another	The device `routes` the data
routesVia	A relationship whereby data from one resource routes via another	The traffic `routesVia` the device
loadBalances	One resource which load balances for others	The load balancer `loadBalances` to servers
resolved	When one resource resolved something for another	DNS server `resolved` IP address
resolves	Represents that one resource uses another to resolve it	FQDN `resolves` to the address
sends	A resource sends some data to another	The application `sends` an SMS message
traverses	Describes when one resource traverses another	The message `traverses` the network
uplinkTo	One resource is up-linked to another	The sensor has an `uplinkTo` the controller

Table 57. Edge types for the **Dependency** *edge labels*

Edge type	Description	Example
dependsOn	A generic dependency between resources	One application `dependsOn` another
runsOn	A resource runs on (and therefore depends on) another resource	The service `runsOn` the host

Table 58. Edge types for the **metaData** *edge labels*

Edge type	Description	Example
metadataFor	A relationship between meta-data and the resource to which it belongs	JSON document `metadataFor` Service

Custom edge types

If none of the default edge types suitably represent a certain relationship, you can define a custom edge type via the `https://localhost:8080/1.0/topology/types/edge` **`POST API`**

Important: A custom edge type needs to be created **before** the observation job passes any edges with the type to the Agile Service Manager topology service.

edgeType
Required

The edgeType name, which has to be unique and **cannot** match the edgeType name of a default edgeType, unless the edgeLabel also matches the corresponding default edge type's edgeLabel parameter.

Restriction: A scenario where both edgeType and edgeLabel match the fields of a default edge type is equivalent to manually creating a default edge types, which is not necessary as default edge types are created implicitly by the topology when needed.

edgeLabel
Required

The edgeLabel of the custom edgeType, which has to be one of the following:

- dataFlow
- dependency
- association
- aggregation

description
Optional (but recommended)

This should be a meaningful description of the type of relationship this edge type represents.

Example:

```
curl -k -X POST --header 'Content-Type: application/json' --header 'Accept:
application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' -d '{
  "edgeType": "connectedTo",
  "edgeLabel": "dataFlow",
  "description": "Default relationship between two devices that exchange data"
}' 'https://localhost:8080/1.0/topology/types/edge'
```

Entity types

The Topology Service allows you to group together resources of the same type using the `entityTypes` property, and identify them in the UI by their icon. Usually a resource has only a single entity type, however, as the property is a *Set*, it is possible for a resource to have more than one entity type. A number of pre-defined entity types are supplied, which are listed in this topic. In addition, you can create additional entity types as required.

Pre-defined entity types

You can access the Swagger documentation for 'entity types' currently in use at the following default link: http://localhost/1.0/topology/swagger#!/Schema/getTypes

An entity type is used to map the resource vertex to an icon, and it also allows for more efficient searching via the **`_type`** query parameter, which can be found in the Swagger documentation at the following default location: http://localhost/1.0/topology/swagger/#!/Resources/getResourceList

Remember: You can create additional custom entity types 'on the fly'.

The following table lists these entity types with links to their icons, if defined.

Table 59. Predefined entity types and icons, where defined

Entity type	Icon
application	
backplane	
bridge	
card	
chassis	
command	
component	
container	
cpu	
database	
directory	
disk	
emailaddress	
event	
fan	
file	
firewall	
fqdn	

Table 59. Predefined entity types and icons, where defined (continued)

Entity type	Icon
group	•••
host	
hsrp	
hub	
ipaddress	
loadbalancer	
location	
networkaddress	
networkinterface	
operatingsystem	OS
organization	
path	
person	
process	
product	
psu	
router	
rsm	
sector	
server	
service	
serviceaccesspoint	
slackchannel	
snmpsystem	SNMP
status	
storage	
subnet	
switch	
tcpudpport	
variable	X=Y

Table 59. Predefined entity types and icons, where defined (continued)

Entity type	Icon
vlan	VLAN
volume	
vpn	VPN
vfr	

Schema REST API

The Topology Service has an API call which will allow you to see all the instantiated entity types in a given topology. The default Swagger location is: http://localhost/1.0/topology/swagger/#!/Schema/getTypes

Important: To retrieve the type name, you must add the `_field=name` query parameter.

REST API

Interactions with the Topology Service take place through a REST API. API calls and Swagger documentation links are listed in this section. Use the Swagger documentation to access more detailed information.

REST API calls and default Swagger links

The REST API calls are grouped together into different sections. If installed on *localhost*, then the Swagger documentation can be accessed at the following link: http://localhost/1.0/topology/swagger/

Composites
The API calls in the composites section of the Merge Service allow you to view which resources have been merged into composites and to create and change them.

http://localhost/1.0/merge/swagger/#/Composites

Groups
The API calls in the **groups** section allow you to create a vertex which represents a group and then to associate other resource vertices with that group.

http://localhost/1.0/topology/swagger/#/Groups

Management artifacts
The Management artifacts calls provide the means to associate non-resource vertices, such as tests, with resource vertices.

http://localhost/1.0/topology/swagger/#/Management_Artifacts

Metadata
Metadata provides the means to store data in a metaData vertex, which can then be associated with a resource vertex using the *metadataFor* edge type.

http://localhost/1.0/topology/swagger/#/Metadata

Resources
The most common calls.

These API calls are used to create, update and delete resource vertices in the topology. It also includes API calls to create, update and delete edges between resources.

http://localhost/1.0/topology/swagger/#/Resources

Note: The topology service has a history model which allows it to retain information on the historical resource properties and edges for 30 days. The Resources methods will take an `_at` query parameter, which will cause them to return what the topology previously looked like at a specific point in time. This allows the UI to visualize the topology as it was in the past.

Rules The API calls in the rules section of the Merge Service allow you to view which merge rules have been defined and to create or update merge rules.

http://localhost/1.0/merge/swagger/#/Rules

Schema
The schema API calls can be used to query the Topology Service for information about the types of entities which exist within the topology.

http://localhost/1.0/topology/swagger/#/Schema

Service info
The Service info API calls include a health check call, and a call to return the current Topology Service version.

http://localhost/1.0/topology/swagger/#/Service_Info

Status The status API provides methods to associate and manipulate the status that is associated with a given resource.

http://localhost/1.0/topology/swagger/#/Types

Tenants
The Tenants API provides a mechanism by which resource, metadata and management artifacts can be made globally readable by multiple tenants.

http://localhost/1.0/topology/swagger/#/Tenants

Types These return information on the entity types which have been instantiated.

Tip: This includes the **`_include_count`** query parameter, which you can use to return a time-stamped count of both the number of types, and number of instances of each type.

http://localhost/1.0/topology/swagger/#/Types

Status (and state)

Resources have specific statuses, and in turn each status has a state of open, clear or closed.

Status

A single status can affect multiple resources, and a single resource can have multiple different statuses. For example, a single *link down* event can generate the status for both the interface resource and the host resource; or a single host could have *CPU* or *disk usage* status in addition to any *link down* status.

Resource status can be viewed in the Netcool Agile Service Manager UI, and can be set or retrieved via the topology service REST API.

You can access the Swagger documentation for 'status' at the following default link: http://localhost/1.0/topology/swagger/#/Status

Important: When modeling resources, you must consider *Status assignment from events*.

Tip: The Topology Service stores the event **Severity**, and nodes in the UI are colored based on severity, which is always one of the following:

- clear
- indeterminate
- information
- warning
- minor
- major
- critical

Take a look at the severity icons in the topology viewer reference topic: Severity icons table

Status assignment

The status of a single resource can be supplied, alongside other resource properties, when creating or updating a resource. Alternatively, an event can generate the status of one or more resources.

Remember: A status **always** has one of the following three states:

- open
- clear
- closed

Status assignment from events

The Event Observer receives events and tries to find matching resources in the topology service. A resource with no match tokens defined will not have events matched to it, but if found, the status of those resources is set from the event data. The assigned status depends on the following event data:

matchTokens
: This property must be used to list any data that can identify (match) the resource.

  Each field may be globally unique, or may be unique within the scope of a composition. In other words, a resource modeled via a composition relationship, such as `partOf`, can be distinguished from other children within the composition using these fields.

  For example, either a globally unique and fully qualified domain name or IP address, or a locally unique interface name (that is, local within the scope of the host), can be used to identify the resource.

partOf composition relationship
: The Event Observer uses composition relationships to match fields that are unique only within the scope of a parent.

  For example, an IP address can be used to find a main node, and an interface name can be used to identify an interface within that main node.

Timestamps

Both vertex and edge graph elements can have multiple timestamps, documented here.

beginTime

The beginTime timestamp records the beginning of a period of time for which the resource was valid, with endTime marking the end of that period.

Tip: A given resource may have multiple begin times in its historic record, and there may be gaps in that record if the resource was offline for periods.

- All resources and historic resources that are representations of the same thing have a distinct beginTime
- Resource beginTime together with endTime is used in historic graph traversals, that is, when the **_at** parameter is supplied. The period during which a resource is valid is defined as:

  ```
  atTime >= beginTime && atTime < endTime
  ```

- A vertex or edge which has the beginTime equal to the endTime can be used to store audit information, such as the provider which deleted a given resource. However, because it takes up zero time it does not form part of the history and is ignored by the above equation.

prevBeginTime

If history exists for a given resource then this property will be set to the beginTime of the most recent historical resource.

changeTime

The changeTime timestamp records when the properties of the element last changed.

Its value may be less than the observedTime, which is updated on a POST or PUT even if no property values have changed.

createTime

The createTime timestamp records when the element was first created in the topology service.

- Historical resources do not store createTime, as it is shared with the anchor.
- This is needed when looking for something older than 30 days, that is, when there is no beginTime this old because the historical resources have timed out.

endTime

The endTime timestamp records when the element was deleted.

- All resources and historic resources that are representations of the same thing have a distinct endTime.
- Resource endTime is used in historic graph traversals, that is, when the **_at** parameter is supplied.
- For current resources, endTime is LONG_MAX. This is sometimes hidden via the REST API.

observedTime

The observedTime timestamp records when the element was last observed, that is, when data was last input to the topology service for the element.

Netcool Agile Service Manager cookbook

The Netcool Agile Service Manager cookbook is a collection of 'recipes' and best-practice guidelines compiled by Netcool Agile Service Manager SMEs, developers and testers. The purpose of this section is to provide you with practical information, such as implementation examples and code samples, that you can use to get started.

Restriction: Recipes provided here must be amended and adjusted to suit your own specific Netcool Agile Service Manager implementation.

Virtual machine recipe

One of the most important goals of Netcool Agile Service Manager is to support the assurance and provisioning of modern IT, network and storage environments. These environments all make extensive use of increasingly nested virtualization technologies that need to be modeled. The following recipe introduces such an IT Virtualization scenario, and describes an OpenStack response that provides a solution.

IT Virtualization

The Netcool Agile Service Manager model of a nested virtualization scenario can extend the traditional types of models provided by other solutions.

This model can represent a multi-domain view of the world that links IT, network, storage, applications and services. In addition, it can incorporate concepts such as OpenStack's Heat Orchestration and Slack collaboration relative to traditional IT resources.

Some of the benefits of this approach are:

To provide additional context

Increasingly, teams are more multi-disciplined and no longer operate in informational or functional silos. For example, network teams may include IT Virtualization specialists.

Such teams need access to additional context when needed in order to answer some of their business-critical questions, such as:

- What storage volume is a VM attached to?
- Which orchestration step realized a network port?
- Who collaborated with whom for a particular incident?
- Which applications and services are supported by a network subnet?
- Which VM instances were shutdown as part of a scale-in activity 1 hour ago?
- What is the impact of removing a given Hypervisor from the environment?
- Which fixed IP addresses have a floating IP address been bound to in the last week?

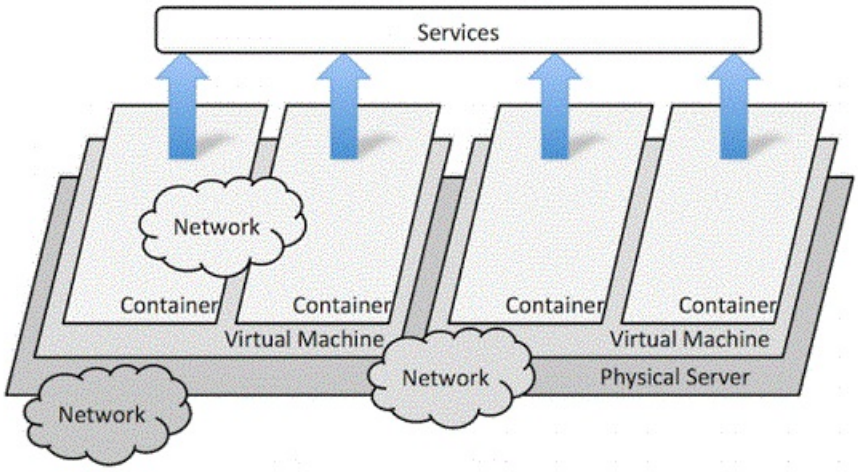
To provide a data-rich base

Value-added services can be bolted onto a base system, provided the information exists, and the system has an architecture that allows for rapid extension.

For example, when building analytics on the topology data, the availability of information such as seasonality can provide additional insights.

The following diagram depicts the nested layers of virtualization, including networking, between these layers and technologies such as Docker and LXC or LXD.

Note: The services exposed can be applications or appear to be traditionally physical services such as network routers, switches, firewalls and load balancers (a key goal of NFV).



OpenStack

OpenStack is a free and open-source platform for cloud computing, typically deployed as an IaaS (Infrastructure-as-a-Service) capability. The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and network resources throughout and between data centers.

OpenStack provides a number of projects, and related services and APIs, that are summarized here, as they speak directly to the need to have a multi-domain view of the environment. For more information, see the OpenStack project navigator and documentation at the following location: https://www.openstack.org/software/project-navigator/

OpenStack **core services** include the following:

Nova *Compute* manages the lifecycle of compute instances in an OpenStack environment.

Neutron

Networking enables network connectivity as a service for other OpenStack services.

Swift *Object Storage* stores and retrieves arbitrarily unstructured data via a REST API.

Cinder
Block Storage provides persistent storage to running instances.

Keystone
Identity provides authentication and authorization services to OpenStack services and a service catalog.

Glance
Image Service stores and retrieves virtual machine disk images for use by Nova.

OpenStack **optional services** include the following:

Horizon
dashboarding

Ceilometer
telemetry

Heat orchestration

Sahara
Elastic Map Reduce

Designate
DNS

Barbican
Key Management

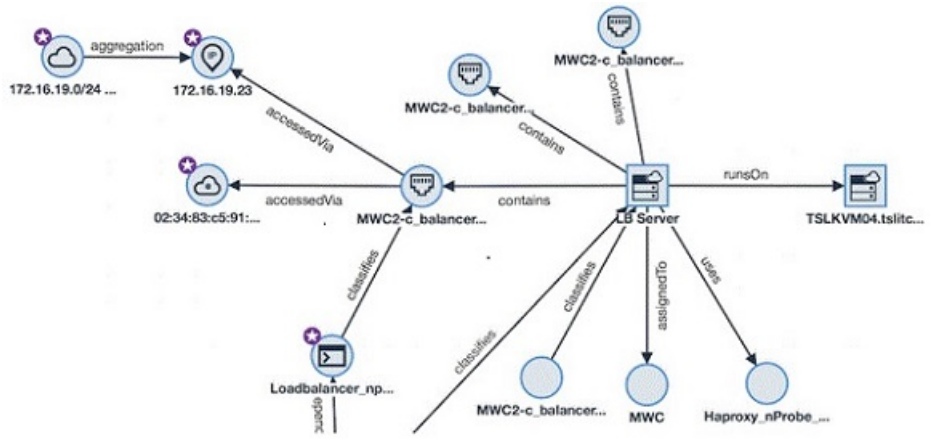
Netcool Agile Service Manager provides an Observer that makes extensive use of OpenStack's core and Heat APIs to build an end-to-end (or multi-domain) model.

IT Virtualization OpenStack scenario

The following example of an OpenStack environment accessed through Netcool Agile Service Manager provides insights into any environment consisting of, for example, a combination of physical services and storage combined with virtualization from VMware (or similar).

IT virtualization

The following figure depicts a Hypervisor running a VM (LB Server), that is assigned to a tenant, has classification data and a number of network interfaces. Each network interface has a MAC address, one or more IP addresses (v4 and/or v6) and each IP address is considered to be part of an IP subnet. It is also related to orchestration data from Heat that helps identify how the instance was created.



Property use guidance

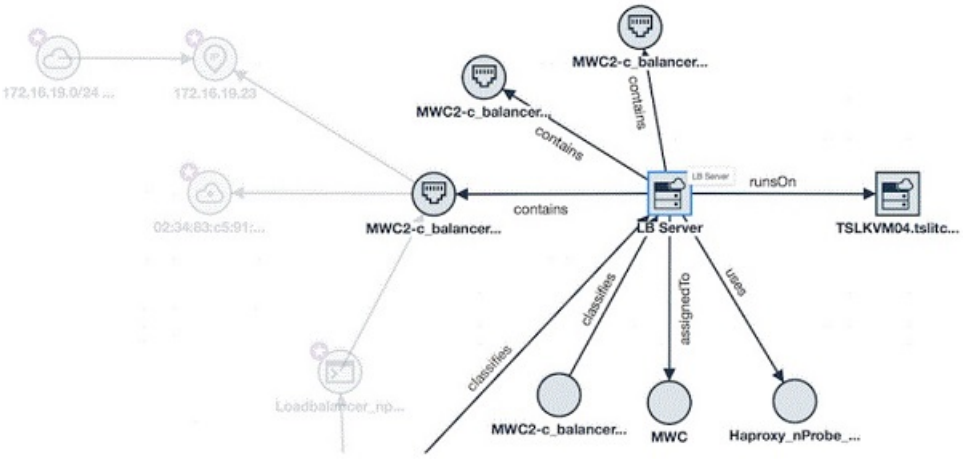
Although the topology service is flexible, you should follow the following guidelines when setting property values to ensure elements are appropriately represented:

- Set `entityType` to one of the pre-defined entity types with an associated icon where possible.
- Set `name` and `description` to be user-friendly values wherever possible.
- Make use of the default generic properties to represent generally applicable characteristics of an element and to provide a basic degree of labeling and filtering. For a list of generic properties, see the following topic: “Properties” on page 258

Model patterns - Part one

Stepping through each of the sections of the example of a multi-domain topology helps to identify reusable patterns.

In the following figure, the Hypervisor 'TSLKVM04' is running a VM 'LB Server'.

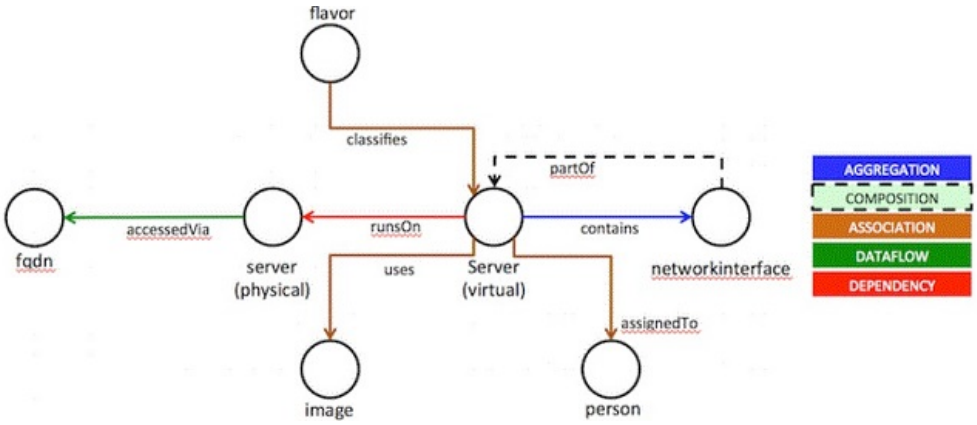


The LB Server in this image is:

- Assigned to the 'MWC' tenant
- Contains three network interfaces

- Uses the Haproxy_nProbe... image (that is, the OpenStack image)
- Is classified as an MWC2-c_balancer (which is the OpenStack flavor)

This means that the following pattern can be identified:



Usage tips

The associations shown for flavor, image and person are optional.

Network interfaces and other components of the device must be associated with it via a partOf relationship. The exception is if an IP or MAC address is known independently of any device, for example flow data would expose those but the device would be unknown.

The FQDN (hostname, short or full DNS name) is associated directly with the server in this case as nothing else is known about the Hypervisor.

Note: This is **not** shown in the topology fragment.

The `partOf` composition is not shown by the topology GUI, but must be created where the relationship between a component and a device is known. This is in addition to relationships such as `contains`, which the GUI will show.

Hypervisor example JSON

```
{
    "_executionTime": 3,
    "createTime": 1501741671066,
    "name": "TSLKVM04",
    "uniqueId": "CYooventry_DC1:MWC/ComputeHost/TSLKVM04",
    "observedTime": 1501776262368,
    "_startedAt": "2017-08-03T06:27:51.066Z",
    "entityTypes": [
        "server"
    ],
    "beginTime": 1501741671066,
    "_id": "KNN6TCGhKyM4MCI6jooGWg",
    "_observedAt": "2017-08-03T16:04:22.368Z",
    "_modifiedAt": "2017-08-03T06:27:51.066Z",
    "_createdAt": "2017-08-03T06:27:51.066Z",
    "changeTime": 1501741671066,
    "matchTokens": [
        "Coventry_DC1:MWC/ComputeHost/TSLKVM04",
        "TSLKVM04"
    ]
}
```

Note: Many of the properties starting with _ are internal (such as timestamps). Also:

- The `uniqueId` is a composite of a number of fields: data center, tenant, classname and name of the Hypervisor because the ID is a highly ambiguous integer.
- The `name` is the name of the instance from OpenStack.
- The `entityType` is set to 'server' to ensure correct classification and icon use.

Virtual machine example JSON

```
{
    "instanceName": "LB server",
    "tenantId": "2f79c691570c4a598be386325ea01da8",
    "launchedAt": 1501741751194,
    "_executionTime": 4,
    "userId": "48c7cd25ad0842be8e9b84390de0e587",
    "imageName": "None Available",
    "availabilityZone": "nova",
    "createTime": 1501741733817,
    "flavorName": "m1.nano",
    "name": "LB server",
    "uniqueId": "1e35c68a-86b0-445f-9741-e581121a0577",
    "serverStatus": "active",
    "observedTime": 1501741752717,
    "_startedAt": "2017-08-03T06:29:12.717Z",
    "entityTypes": [
        "server",
        "vm"
    ],
    "beginTime": 1501741752717,
    "flavorId": "42",
    "vmState": "active",
    "_id": "3nDmTkKfrvNhZinSDCYHDw",
    "_observedAt": "2017-08-03T06:29:12.717Z",
    "createdAt": 1501741733000,
    "_modifiedAt": "2017-08-03T06:29:12.717Z",
    "_createdAt": "2017-08-03T06:28:53.817Z",
    "changeTime": 1501741752717,
    "matchTokens": [
        "1e35c68a-86b0-445f-9741-e581121a0577",
        "LB server"
    ]
}
```

Note: Many of the properties starting with _ are internal (such as timestamps). Also:

- The `uniqueId` in this case is the UUID of the instance from OpenStack.
- The `name` is the name of the instance from OpenStack.
- The `entityType` is set to 'server' and 'vm' to ensure correct classification and icon use.
- The `isVirtual` boolean is set to true.

Network interface example JSON

```
{
    "_executionTime": 3,
    "isAdminStateUp": true,
    "createTime": 1501741717674,
    "name": "aab47c85-3110-401a-8bad-960b7c4bcd7b",
    "uniqueId": "aab47c85-3110-401a-8bad-960b7c4bcd7b",
    "observedTime": 1501741718855,
    "_startedAt": "2017-08-03T06:28:38.855Z",
    "entityTypes": [
```

```
            "networkinterface"
        ],
        "beginTime": 1501741718855,
        "isPortSecurityEnabled": true,
        "_id": "kkKpDH6yLn7cmNVXOddImg",
        "_observedAt": "2017-08-03T06:28:38.855Z",
        "_modifiedAt": "2017-08-03T06:28:38.855Z",
        "_createdAt": "2017-08-03T06:28:37.674Z",
        "changeTime": 1501741718855,
        "matchTokens": [
            "aab47c85-3110-401a-8bad-960b7c4bcd7b"
        ]
    }
```

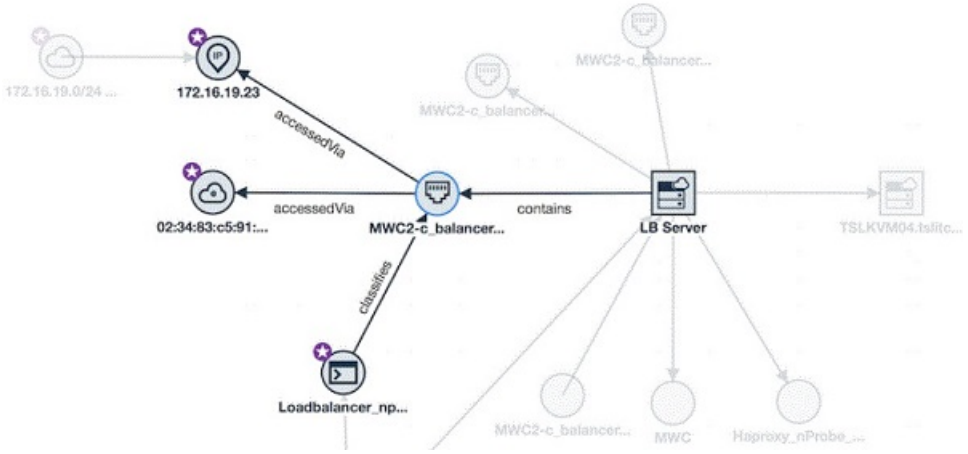
Note: As with the LB Server data, many of the properties are internal. Also:

- The `uniqueId` in this case is the UUID of the instance from OpenStack.
- The `name` is the name of the instance from OpenStack.
- The `entityType` is set to 'networkinterface' to ensure correct classification and icon use.
- The type of the interface (such as `ifType` from ITNM) is unknown.

Model patterns - Part two

Remember: We are stepping through each of the sections of the example of a multi-domain topology to identify reusable patterns.

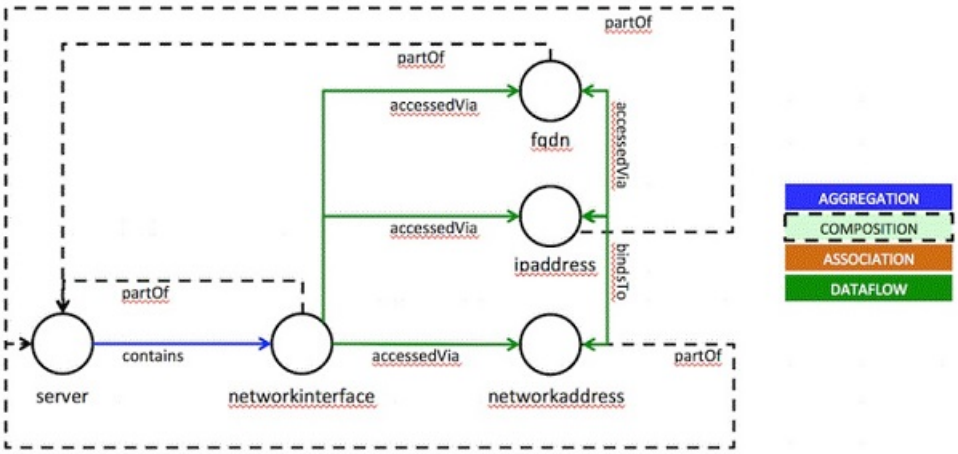
In the following figure, the LB Server VM contains a network interface.



The network interface in this image is:

- Accessed via an IP address and a MAC address
- Classified by a Heat orchestration resource

This means that the following pattern can be identified:



Usage Tips

The Heat orchestration element and relationship is optional. Such things should be added if known to provide additional context.

The network interface must be contained by and `partOf` the device. `Contains` is more fine-grained and may reference intermediate cards (for example) within the containment hierarchy of the device, such as

```
json device--contains-->card--->contains--->port
```

If an IP address, FQDN or MAC address is known to be related to a specific device, then they must be associated with the device via a `partOf` relationship *in addition* to the `accessedVia` relationship.

If an IP address, FQDN or MAC address are known independently of a device, then no `partOf` relationship from them is necessary.

If an IP address is known to resolve to an FQDN, then relationships between them should be created to depict that one resolves to another and one accesses another (`accessedVia` shown in the example).

IP address example JSON

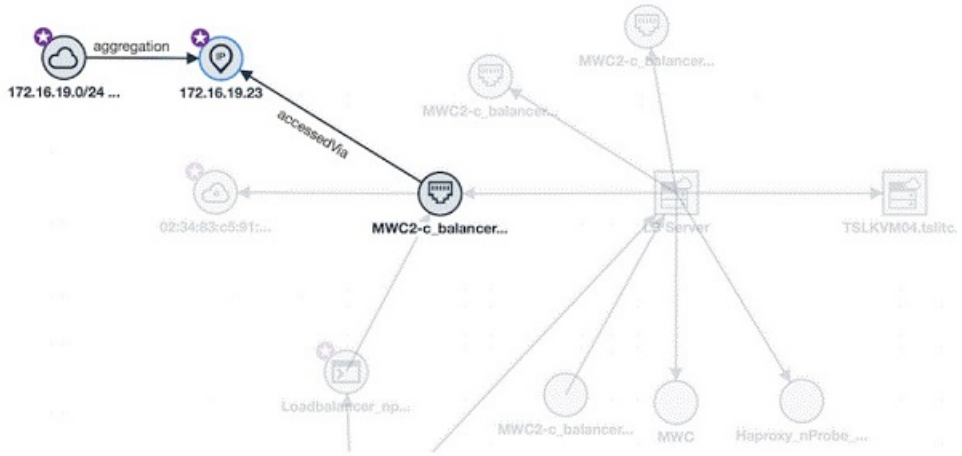
```
{
    "_executionTime": 3,
    "createTime": 1501741717656,
    "name": "172.24.4.5",
    "uniqueId": "172.24.4.5",
    "ipNumber": 2887255045,
    "addressSpace": "Coventry_DC1:MWC",
    "observedTime": 1501741731565,
    "_startedAt": "2017-08-03T06:28:51.565Z",
    "entityTypes": [
        "ipaddress"
    ],
    "beginTime": 1501741731565,
    "version": "IPv4",
    "_id": "jPLc72DU-UvPeTQE_7YdPQ",
    "_observedAt": "2017-08-03T06:28:51.565Z",
    "_modifiedAt": "2017-08-03T06:28:51.565Z",
    "_createdAt": "2017-08-03T06:28:37.656Z",
    "changeTime": 1501741731565,
    "matchTokens": [
        "172.24.4.5",
        "IPv4:2887255045"
    ]
}
```

Note:

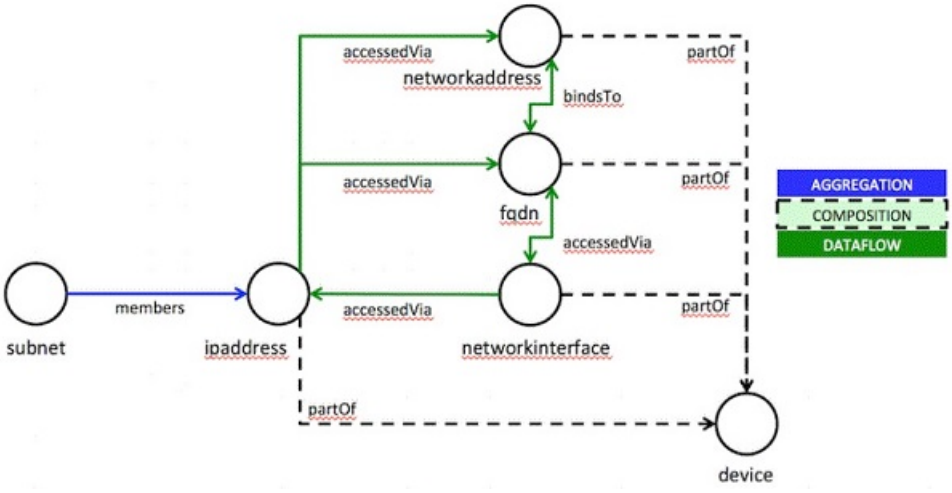
- The `uniqueId` is the IP address itself. This is an RFC1918 IP address and so it **must** be qualified with an address space to disambiguate it from other instances of the same IP address. Similar precautions should be used with MAC addresses, which can be ambiguous.
- The `protocol` reflects whether the IP address is an IPv4 or IPv6 address.
- The `ipNumber` is a numeric representation of the IPv4 or IPv6 address. A Java BigInteger has the precision to represent IPv6 addresses.

Model patterns - Part three

Remember: We are stepping through each of the sections of the example of a multi-domain topology to identify reusable patterns.



This means that the following pattern can be identified:



Usage Tips

The IP subnet should aggregate any IP addresses known to be in it.

Elements accessed via an IP should be related to it accordingly, e.g. a network interface and/or service or process.

If an FQDN is known to resolve to an IP address (and vice-versa), then they should be related.

If a MAC address is known to bind to an IP address and vice-versa, they should be related.

If an IP address, MAC address or FQDN are known to relate to a device, they should be considered `partOf` it; otherwise they are independent.

IP subnet example JSON


```
{
  "uniqueId": "c009ff59-13b9-48dc-8863-cd0c75070d99",
  "name": "172.24.4.0/24 (public-subnet)",
  "entityTypes": [
    "subnet"
  ],
  "matchTokens": [
    "172.24.4.0/24 (public-subnet)",
    "172.24.4.0/24",
    "c009ff59-13b9-48dc-8863-cd0c75070d99"
  ],
  "_id": "u0sjaHcumtg5A4DRl1fyAQ",
  "_references": [
    {
      "_id": "9duzx1-3o52g-ys5-47si0",
      "_edgeType": "contains",
      "_label": "aggregation",
      "_fromId": "u0sjaHcumtg5A4DRl1fyAQ",
      "_toId": "lSJQs8JmYzDQ-_wUOfvJbg",
      "_fromUniqueId": "c009ff59-13b9-48dc-8863-cd0c75070d99",
      "_toUniqueId": "172.24.4.12",
      "createTime": 1501838680418,
      "_observedAt": "2017-08-04T09:24:40.418Z",
      "_createdAt": "2017-08-04T09:24:40.418Z",
      "beginTime": 1501838680418,
      "_startedAt": "2017-08-04T09:24:40.418Z",
      "observedTime": 1501838680418
    },
  ],
  "_executionTime": 11,
  "_modifiedAt": "2017-08-04T09:24:40.356Z",
  "isDhcpEnabled": false,
  "dnsNames": "None Available",
  "_observedAt": "2017-08-04T09:24:40.356Z",
  "gatewayIp": "172.24.4.1",
  "_startedAt": "2017-08-04T09:24:40.356Z",
  "observedTime": 1501838680356,
  "changeTime": 1501838680356,
  "ipVersion": "V4",
  "createTime": 1501838680356,
  "_createdAt": "2017-08-04T09:24:40.356Z",
  "cidr": "172.24.4.0/24",
  "networkId": "3e2b5d07-653a-4fc8-8224-45801d9d113f",
  "beginTime": 1501838680356,
  "allocationPools": "172.24.4.2-to-172.24.4.254"
}
```


Note:

- The `uniqueId` in this case is the UUID of the subnet from OpenStack.
- The `name` in this case is set to CIDR notation plus the ID of the subnet.
- The `entityType` is set to subnet to ensure appropriate classification and icon usage.
- Some example relationships are shown: For example, an IP address that is part of the subnet is visible, and the subnet's use of an allocation pool is also depicted.

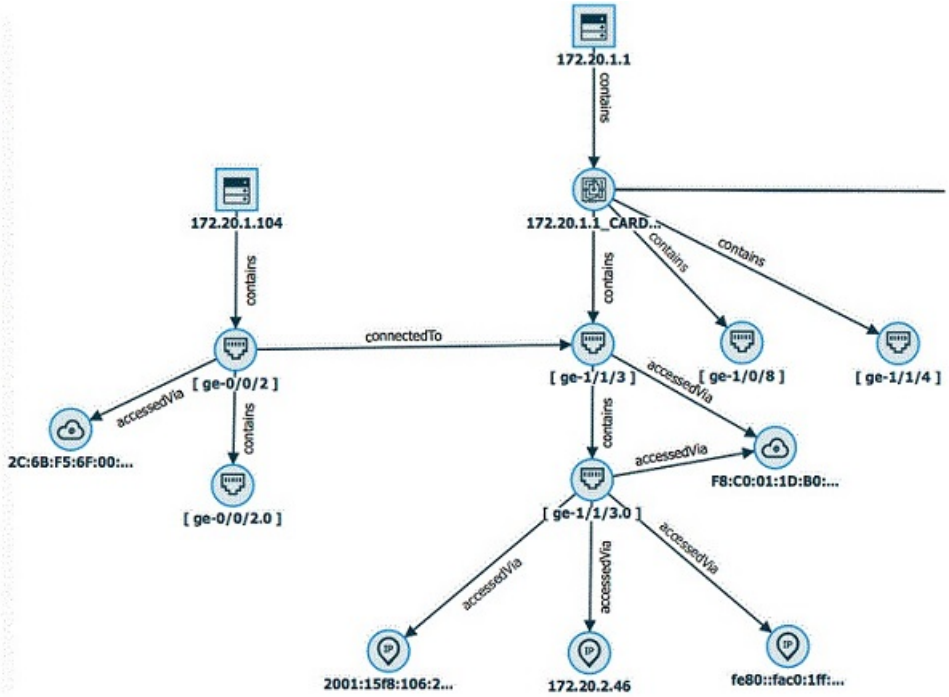
Physical device recipe

The following example of an ITNM environment accessed through Netcool Agile Service Manager provides insights into any environment that makes use of physical network devices.

Network physical devices

The following figure depicts two physical devices, 172.20.1.104 and 172.20.1.1, which are connected to each other.

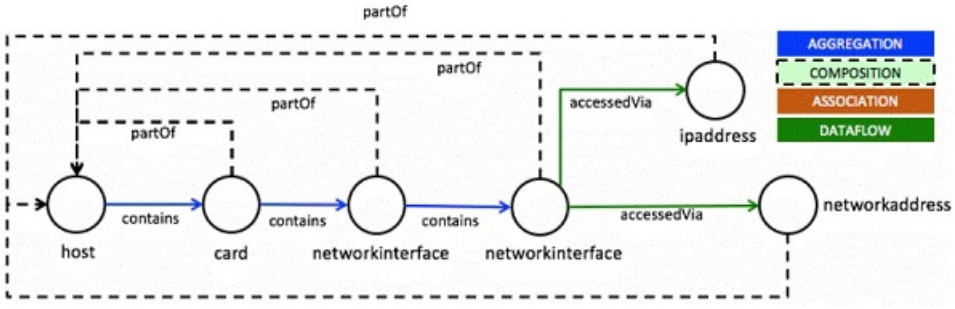
Server 172.20.1.1 has three Gigabit Ethernet ports on the same card, one of which has a sub-interface with an `ifName` of 'ge-1/1/3.0'. That sub-interface shares the same MAC address as its physical parent and has two IPv6 addresses and one IPv4 address associated with it.



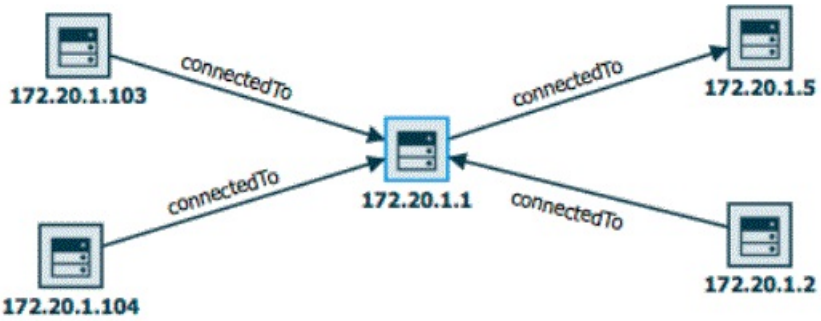
Outline pattern

The topology service has an edge type of `partOf`, which is a member of the composition edge label family. See the “Edge labels” on page 260 topic for more information.

The following image illustrates how this relationship is used to tie the card, interfaces and network addresses to the hosting device.



The `partOf` relationship is not shown as an explicit edge in the toplogy UI. However, it is used to determine which resources should be hidden in the host-to-host view, which in the context of this scenario would show just the hosts 172.20.1.104 and 172.20.1.1, as well as the `connectedTo` edge between them, as depicted in the following part of the image.



Host example JSON

The following JSON extract is an example of the properties that you might choose to include when creating a host vertex. The properties which start with an underscore character are aliases for some of the read-only generic properties, and there are also a few read-write generic properties, such as `name` and `uniqueId`; however, the majority of the properties in this example are free from 'User' properties.

```
{
  "uniqueId": "NCOMS:172.20.1.1",
  "name": "172.20.1.1",
  "entityTypes": [
    "host"
  ],
  "_createdAt": "2017-03-03T16:02:40.845Z",
  "_observedAt": "2017-03-03T16:04:18.636Z",
  "_id": "y7EXOKrHud21CWySCyMsBg",
  "_href": "/1.0/topology/resources/y7EXOKrHud21CWySCyMsBg",
  "_nodeType": "resource",
  "_executionTime": 4,
  "_modifiedAt": "2017-03-03T16:02:40.845Z",
  "matchTokens": [
    "sbk-pe1-jrmx80.southbank.eu.test.lab"
  ],
  "sysObjectId": "1.3.6.1.4.1.2636.1.1.1.2.90",
  "entityChangeTime": "2017-03-03T14:13:17.000Z",
  "className": "JuniperMSeries",
  "services": "datalink(2) network(3)",
  "_startedAt": "2017-03-03T16:02:40.845Z",
  "manual": 0,
```

```
  "cdmAdminState": 0,
  "cdmType": 2,
  "sysDescription": "Juniper Networks, Inc. mx5-t internet router,
kernel JUNOS 15.1F4.15, Build date: 2015-12-23 20:50:37 UTC
Copyright (c) 1996-2015 Juniper Networks, Inc.",
  "sysName": "sbk-pe1-jrmx80.southbank.eu.test.lab",
  "sysLocation": "The Mad Hatter Hotel 3-7 Stamford St
London SE1 9NY UK,-0.10499474,51.50711477",
  "interfaceCount": 73,
  "entityCreateTime": "2017-03-03T14:13:17.000Z",
  "isIpForwarding": "forwarding",
  "accessIPAddress": "172.20.1.1",
  "entityDiscoveryTime": "2017-03-03T14:11:09.000Z",
  "sysContact": "williamking@uk.ibm.com",
  "_tenantIds": [
    "Moa1dcmKHfx3dlyJnGm6JQ"
  ],
  "accessProtocol": "IPv4"
}
```

Note: Some of the generic properties in this example are the following:

- The `uniqueId` in this case is a string, which uniquely identifies this resource to ITNM.
- The `name` is the name of the host and will be used in the UI.
- The `entityType` is set to 'host'.
- The `matchTokens` contains the FQDN name of the host.

Network interface example JSON

```
{
  "uniqueId": "NCOMS:172.20.1.1[ ge-1/1/3.0 ]",
  "name": "[ ge-1/1/3.0 ]",
  "entityTypes": [
    "networkinterface"
  ],
  "_createdAt": "2017-03-03T16:03:03.889Z",
  "_observedAt": "2017-03-03T16:03:03.939Z",
  "_id": "v8aFVG6JoigYxTrlFSDkLw",
  "_href": "/1.0/topology/resources/v8aFVG6JoigYxTrlFSDkLw",
  "_nodeType": "resource",
  "_executionTime": 5,
  "operationalStatus": "started",
  "ifIndex": 537,
  "ifAdminStatus": "up",
  "_modifiedAt": "2017-03-03T16:03:03.940Z",
  "ifType": 53,
  "matchTokens": [
    "ge-1/1/3.0",
    "ifEntry.537"
  ],
  "ifName": "ge-1/1/3.0",
  "ifTypeString": "propVirtual",
  "connectorPresent": "false",
  "_startedAt": "2017-03-03T16:03:03.939Z",
  "speed": 1000000000,
  "mtu": 1500,
  "accessIpAddress": "172.20.2.46",
  "operationalDuplex": "FullDuplex",
  "promiscuous": false,
  "physicalAddress": "F8:C0:01:1D:B0:13",
  "ifDescription": "ge-1/1/3.0",
  "ifOperStatus": "up",
  "_tenantIds": [
```

```
    "Moa1dcmKHfx3dlyJnGm6JQ"
  ],
  "accessProtocol": "IPv4"
}
```

Note: Some of the generic properties in this example are the following:

- The `uniqueId` in this case is a string, which uniquely identifies this resource to ITNM.
- The `name` is the name of the host and will be used in the UI.
- The `entityType` is set to 'networkinterface'.
- The `matchTokens` contains both the `ifName` and a string denoting the `ifIndex` in the `ifEntry` table.

XML Gateway reference

Agile Service Manager resource status can be generated from Netcool/OMNIbus events. The gateway must be configured to post XML events to the Event Observer.

Prerequisites

Remember: For up-to-date information on the version of the XML Gateway required, see "Software requirements" on page 10.

Location

The default $NCHOME install location is `/opt/IBM/tivoli/netcool` and the $OMNIHOME install location is `$NCHOME/omnibus`.

Tip: Export OMNIHOME as an environment variable, as it is repeatedly used in the scripts.

Standard gateway configuration

You must create a `$NCHOME/etc/omni.dat` entry for the gateway (which in these examples is assumed to be **G_ASM**):

```
[G_ASM]
 {
 Primary: nasm-test1 4300
 }
```

Run **`$NCHOME/bin/nco_igen`**

Generate a key file with nco_keygen.

Minimum XML gateway configuration requirements

For the XML gateway to post XML events to the Event Observer, you must edit the following files as a minimum:

XML Gateway properties file

If this file does not exist, you must create it in the `$OMNIHOME/etc` directory.

For example, the XML Gateway properties file for a gateway called G_ASM would be `$OMNIHOME/etc/G_ASM.props`

You define a number of properties, such as the name of the Netcool/OMNIbus Object Server, in the XML Gateway properties file.

You also reference the transformers XML file and the transport properties file here.

XML Gateway transport properties file
The file name of the XML Gateway transport properties file must match the one referenced in the XML Gateway properties file.

Here you define as a minimum the URL of the Event Observer to which XML events are posted, the batch header and footer, the maximum number of events in a single batch, the maximum waiting period before sending the events, and access to the HTTPS (TLS) truststore.

Default location and name: `$OMNIHOME/java/conf/asm_httpTransport.properties`

XML Gateway transformer XML file
The file name of the XML Gateway transformer XML file must match the one referenced in the XML Gateway properties file.

Here you define as a minimum the URL of the Event Observer to which XML events are posted.

Default location and name: `$OMNIHOME/java/conf/asm_Transformers.xml`

Additional information

For more information on configuring the XML gateway, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/gateways/xmlintegration/wip/concept/xmlgw_intro.html

For additional gateway configuration information, see the following IBM developerWorks discussion: https://developer.ibm.com/answers/questions/256154/how-is-the-xml-message-bus-probe-and-gateway-confi.html

Gateway properties file

You create and/or edit the XML Gateway properties file: `$OMNIHOME/etc/<your gateway>.props` and then define at least the following properties:

- The name of the Netcool/OMNIbus ObjectServer
- The name of the transport properties file
- The name of the transformer XML file

The following sample code is for a `$OMNIHOME/etc/G_ASM.props` gateway properties file, retrieving data from the `AGG_V` ObjectServer via the `G_ASM` gateway.

```
# Standard properties
Gate.Reader.Server  : 'AGG_V'

# Properties defining XML messages over HTTP
Gate.MapFile:                    '$OMNIHOME/gates/xml/asm_xml.map'
Gate.StartupCmdFile:             '$OMNIHOME/gates/xml/xml.startup.cmd'
Gate.Reader.TblReplicateDefFile: '$OMNIHOME/gates/xml/asm_xml.reader.tblrep.def'
Gate.XMLGateway.TransformerFile: '$OMNIHOME/java/conf/asm_transformers.xml'
Gate.XMLGateway.TransportFile:   '$OMNIHOME/java/conf/asm_httpTransport.properties'
Gate.XMLGateway.TransportType:   'HTTP'

# The event observer requires the timestamp in this format, including the timezone
Gate.XMLGateway.DateFormat      : 'yyyy-MM-dd\'T\'HH:mm:ssZ'

# To flush events to the gateway from the object server at 5s intervals, use this
Gate.Reader.IducFlushRate : 5

#####################################################
# Security credentials required for the proxy service
# For full FIPS compliance, alter the java.security file as per the Omnibus documentation
#####################################################
# This algorithm must be AES_FIPS
```

```
ConfigCryptoAlg: 'AES_FIPS'
# Secure key file generated using nco_keygen
ConfigKeyFile:   '/opt/IBM/netcool/core/omnibus/etc/crypto.key'
```

Important: Do **not** use the $OMNIHOME variable in ConfigKeyFile.
Example mapping (minimum fields required)

Note: The name of the gateway map file must match the one specified by the `Gate.MapFile` property in the gateway properties file.

```
CREATE MAPPING StatusMap
(
  'Agent'          = '@Agent',
  'AlertGroup'     = '@AlertGroup',
  'Class'          = '@Class',
  'Customer'       = '@Customer',
  'EventId'        = '@EventId',
  'Identifier'     = '@Identifier',
  'LastOccurrence' = '@LastOccurrence',
  'LocalPriObj'    = '@LocalPriObj',
  'LocalRootObj'   = '@LocalRootObj',
  'Manager'        = '@Manager',
  'Node'           = '@Node',
  'NodeAlias'      = '@NodeAlias',
  'ServerName'     = '@ServerName',
  'ServerSerial'   = '@ServerSerial',
  'Severity'       = '@Severity',
  'StateChange'    = '@StateChange',
  'Summary'        = '@Summary',
  'Type'           = '@Type'
);
```

Gateway transport properties file

Note: The name of the gateway transport properties file must match the one specified by the `Gate.XMLGateway.TransportFile` property in the gateway properties file.
The gateway transport properties file (in these examples `$OMNIHOME/java/conf/asm_httpTransport.properties`) must specify at least the following properties, as shown in the example:

- The user's authentication credentials.
- The batch header and footer, exactly as shown in the example.
- The size and flush time, which specify the maximum number of events in a single XML batch file, and the maximum wait, in seconds, before sending the events.
- The proxy service username and password. Encrypt the proxy service password (and optionally the username):

  ```
  nco_aes_crypt -c AES_FIPS -k /opt/IBM/netcool/core/omnibus/etc/crypto.key <password>
  ```

- Add the username and password to the asm_httpTransport.properties file, for example:

  ```
  username=asm
  password=@44:9WxiH51VqMNHNYOLvoShaXO01KwBLqXtGqtB/ZGCYPo=@
  ```

  Tip: You only edit the java security file for FIPS compliance.
- For gateway access to the truststore, you need to complete the following steps:
  1. Create a truststore from the ASM CA certificate, and copy it to the Netcool/OMNIbus host (if different).

```
keytool -import \
        -alias asm-ca \
        -file $ASM_HOME/security/asm-ca.crt \
        -keystore gateway_truststore.jks \  <---- this file needs to go in the gw config
        -storetype JKS \
        -noprompt
```

While running the command, you are prompted for a password.

2. Add the truststore and password to the Gateway transport properties file. When completed, the gateway transport properties file should contain the following:
   - `trustStore=/fullPath/gateway_truststore.jks`
   - `trustStorePassword={passwordGivenInPreviousStep}`

   In the following example you are copying `gateway_truststore.jks` from the Agile Service Manager server:

```
$ grep ^trust /opt/IBM/netcool/core/omnibus/java/conf/asm_httpTransport.properties
trustStore=/opt/ibm/netcool/asm/security/gateway_truststore.jks
trustStorePassword=changeit
```

Optionally, you can also define:

- The timeout, which is the amount of time in seconds that an http client waits before aborting the connection.
- The retry limit, which is the number of times a http client tries to connect.
- The retry wait time, which is the amount of time (in seconds) an http client waits before attempting to reconnect.

Gateway transformer XML file

The gateway transformer XML file (`$OMNIHOME/java/conf/asm_Transformers.xml`) must specify at least the URL of the Event Observer (`endpoint`), to which XML events are posted.

Note: The name of the gateway transformer XML file must match the one specified by the `Gate.XMLGateway.TransformerFile` property in the gateway properties file.
In the following example, the `your host` part of the URL specified in `endpoint` will be specific to your installation.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformers
    xmlns:tns="http://item.tivoli.ibm.com/omnibus/netcool/transformer"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <tns:transformer name="netcoolEvents" type="northbound"
endpoint="https://<your-asm-host>/1.0/event-observer/netcool/list"
className="com.ibm.tivoli.netcool.integrations.transformer.XSLTTransformer">
        <tns:property name="xsltFilename" type="java.lang.String"
value="${OMNIHOME}/java/conf/netcoolstripxmlheader.xsl"
description="XSLT file for converting Netcool events to NC events"/>
    </tns:transformer>
</tns:transformers>
```

State and status derived from Netcool/OMNIbus

The Event Observer derives the status of resources from individual fields of the event.

Table 60. General event state rules

State	Meaning	Netcool/OMNIbus event mapping
closed	An active issue, may require attention	Active event with Severity > 0

Table 60. General event state rules (continued)

State	Meaning	Netcool/OMNIbus event mapping
open	Current state, working as expected	Cleared event with Severity = 0
clear	No longer relevant	Deleted event

Table 61. Use of Netcool/OMNIbus alerts.status event fields by Agile Service Manager

alerts.status fields	Use by Agile Service Manager
Agent	Provider name for events generated from Agile Service Manager
AlertGroup	Type of Agile Service Manager event
Class	45111 for Agile Service Manager events (should be mapped in alerts.conversions)
Customer	TenantId for events generated from Agile Service Manager
EventId	Status [type] for events generated from Agile Service Manager
Identifier	Determines the type of status, populating the status field
LastOccurrence	Used for the observedTime of open events
LocalPriObj	Resource lookup
LocalRootObj	Resource lookup
Manager	Observer name for events generated from Agile Service Manager
Node	Resource lookup
NodeAlias	Resource lookup
ServerName	Used to generate the unique eventId
ServerSerial	Used to generate the unique eventId
Severity	Severity 0 events represent a clear state
StateChange	Used for the observedTime of clear events
Summary	Used for the status description, shown in the UI
Type	Only Type 1 (Problem), Type 13 (Information) and Type 20 (ITMProblem) events are processed. All others are ignored.

Table 62. Netcool/OMNIbus event data mapped onto Topology Service status

Topology Service status field	Netcool/OMNIbus source
description	alerts.status Summary
eventId	<ServerName>/<ServerSerial>
eventManager	"netcool"
observedTime	closed - time event received by observer clear - alerts.status StateChange open - alerts.status ObservedTime
severity	alerts.status Severity

Table 62. Netcool/OMNIbus event data mapped onto Topology Service status (continued)

Topology Service status field	Netcool/OMNIbus source
state	closed - deleted events clear - Severity 0 events open - none of the above
status	alerts.status Identifier

Filtering events

Note: The name of the table replication definition file must match the one specified by the `Gate.Reader.TblReplicateDefFile` property in the gateway properties file.
To improve performance and prevent unnecessary events from being displayed in the topology viewer, you can filter events by type, and then refine these further by extracting only the fields of interest.

For example, you may want to include only the following event types:

- Problem (Type 1)
- Information (Type 13)
- ITMProblem (Type 20)

You may also want to remove all Netcool/OMNIbus self-monitoring events, that is, class 99999 events.

To include only problem (type 1), information (type 13), and ITMProblem (type 20) event types, and exclude Netcool/OMNIbus self-monitoring events (class 99999), use the following code:

```
asm_xml.reader.tblrep.def: |-
REPLICATE ALL FROM TABLE 'alerts.status'
USING MAP 'StatusMap'
FILTER WITH 'Type IN (1, 13, 20) AND Class != 99999';
```

Probe for Message Bus reference

The Netcool/OMNIbus Message Bus probe must be configured to receive status from Agile Service Manager in JSON format via HTTP, and generate corresponding events in the Netcool/OMNIbus Event Viewer. These events are then fed back to the Agile Service Manager via the Netcool/OMNIbus XML Gateway, which updates the Agile Service Manager status via the Event Observer with the eventId.

Prerequisites

Location

The default OMNIHOME install location is `/opt/IBM/tivoli/netcool/omnibus`

Probe for Message Bus configuration requirements

Tip: You can use the `topology_service_probe_list.sh` script (run without credentials) to list configured probes.

For the probe to receive status from Agile Service Manager, you must edit the following files as a minimum:

Probe properties file

Create and edit the probe property file.

In the following example, a non-default property file is used, which requires the -propsfile option when running the probe.

```
cd $OMNIHOME/probes/linux2x86/
cp message_bus.props asm_message_bus.props
```

Edit the asm_message_bus.props file as in the following example:

```
# Tell the probe to expect json over REST
MessagePayload  : 'json'
TransformerFile : ''
TransportFile   : '$OMNIHOME/java/conf/probe_httpTransport.properties'
TransportType   : 'HTTP'

# Tell the probe how to parse the json payload, such that each member
of its variable-length
# _status array is processed as a separate message, with top-level
properties also included
MessageHeader                   : 'json'
MessagePayload                  : 'json._status'

# standard probe properties
MessageLog                      : '$OMNIHOME/log/asm_probe.log'
RulesFile                       : '$OMNIHOME/probes/linux2x86/
asm_message_bus.rules'
```

Probe transport file

Create and edit the probe transport file.

The name of the probe transport file must match the name given in the probe properties, in this example 'probe_httpTransport.properties'

Create a new file if necessary:

```
cd $OMNIHOME
cp java/conf/httpTransport.properties java/conf/probe_httpTransport.properties
```

This file needs to specify at least the URL of the probe, where it will accept JSON status as input; for example:

```
serverPort=http:18080
```

This port number is required when registering the probe URL.

Probe rules file

You use the supplied probe rules file (`$ASM_HOME/integrations/omnibus/asm_message_bus.rules`).

The name of the probe rules file must match the name given in the probe properties, which in this example is 'asm_message_bus.rules'

The probe rules transform the input into events suitable for the Netcool/OMNIbus alerts.status table. The name of the file must be given as a probe property or command line option.

Create a new file if necessary, by copying and editing the supplied file:

```
cd $OMNIHOME/probes/linux2x86/
cp message_bus.rules asm_message_bus.rules
```

Registering a probe URL to which status is exported

Important ICP Note: Do not configure an event sink when using the ICP version of Agile Service Manager, as this has been configured during installation.
To register a probe URL as an event sink, you run the following command:

```
cd $ASM_HOME/bin
topology_service_probe_register.sh -url http://{probeHost}:{probePort}
```

For example:

```
topology_service_probe_register.sh -url http://probe-host.ibm.com:18080
```

Tip: Docker comes with a default bridge network, docker0. This allows access to the docker host from within the docker container. By default, the docker host is available as `172.17.0.1`, so an omnibus probe running on port `18080` of the Docker host could be configured for use via:

```
topology_service_probe_register.sh -url http://172.17.0.1:18080
```

The Agile Service Manager observer framework automatically emits status if an ASM_EVENT_SINK management artifact (stored in the topology service graph) has been configured, as depicted in the following example. The URL must include the probe's serverPort.

```
{
     "keyIndexName": "<your probe on your-system>",    <- any unique name
     "entityTypes": [
         "ASM_EVENT_SINK"                             <- must be exactly this
     ],
     "tags": [
         "ASM_OBSERVER_CONFIG"                        <- must be exactly this
     ],
     "url": "http://<your.hostname>:18080"   <- must match the probe's
configured serverPort
 }
```

Additional information

For more information on configuring the probe, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/message_bus/wip/concept/messbuspr_intro.html

For information on using the probe and the gateway as a single implementation, see the following section in the Netcool/OMNIbus Knowledge Center: https://www.ibm.com/support/knowledgecenter/en/SSSHTQ/omnibus/probes/message_bus/wip/concept/messbuspr_integration_intro.html

Example probe rules file

The following is an example of a rules file.

asm_message_bus.rules

```
if( match( @Manager, "ProbeWatch" ) )
{
    switch(@Summary)
    {
    case "Running ...":
            @Severity = 1
            @AlertGroup = "probestat"
            @Type = 2
    case "Going Down ...":
            @Severity = 5
            @AlertGroup = "probestat"
            @Type = 1
    case "Start resynchronization" | "Finish resynchronization":
            @Severity = 2
            @AlertGroup = "probestat"
            @Type = 13
```

```
    case "Connection to source lost":
            @Severity = 5
            @AlertGroup = "probestat"
            @Type = 1
    default:
            @Severity = 1
    }
    @AlertKey = @Agent
    @Summary = @Agent + " probe on " + @Node + ": " + @Summary
}
else
{
    ################################################
    # Input from ASM
    #
    # guaranteed json fields: uniqueId, observerName, providerName, status, state
    # optional json fields: name, description, severity, uuid
    # guaranteed http headers: X-TenantID
    #
    ################################################

    # These 4 fields are used to match up the event to the ASM status, and must
not be changed
    @EventId        = $status
    @Manager        = $observerName
    @Agent          = $providerName
    @NodeAlias      = $uniqueId

    # This is a user-friendly string identifying the resource
    @Node           = $uniqueId
    if ( exists($name) )
    {
        @Node=$name
    }

    if ( exists($uuid) )
    {
        @LocalNodeAlias=$uuid
    }

    @AlertGroup = "ASM Status"
    if ( exists($eventType) )
    {
        @AlertGroup=$eventType
    }

    switch($state)
    {
    case "open":
        @Type = 1
    case "clear":
        @Type = 2
    case "closed":
        @Type = 2
    default:
        @Type = 1
    }

    @Severity=1
    if ( exists($severity) )
    {
        switch($severity)
        {
        case "clear":
            @Severity = 1
        case "indeterminate":
            @Severity = 1
        case "warning":
            @Severity = 2
        case "minor":
            @Severity = 3
```

```
        case "major":
            @Severity = 4
        case "critical":
            @Severity = 5
        default:
            @Severity = 1
        }
    }

    @Summary=$status
    if ( exists($description) )
    {
        @Summary=$description
    }

    # Use the status timestamp so that the event isn't older than the topology status
    if ( exists($observedTime) && int($observedTime) > 0 )
    {
            # The Object Server uses seconds, whereas ASM uses milliseconds
            $seconds = regreplace($observedTime, "(.*?)\d\d\d$", "\1")
            @LastOccurrence = $seconds
    }

    # Strip out the headers of interest. These are enclosed in square
    # brackets (basically string reprsentations of arrays)
    @Customer = extract( $(MESSAGE.META.X-TenantID), "^\[(.*)\]$" )

    @AlertKey=$uniqueId + "->" + $status + "->" + @Manager
    @Identifier=@AlertKey + @Type

    @Class = 45111
}
```

Topology viewer reference

This reference topic describes the Netcool Agile Service Manager UI screen elements and associated functionality.

Navigation toolbar

The navigation toolbar is displayed at the top of the Topology Viewer and provides access to the following functionality, or information.

Resource Search

The seed resource of the topology visualization.

You define the seed resource around which a topology view is rendered by searching for a seed in the resource search. As you type in a search term related to the resource that you wish to find, such as name or server, a drop-down list is displayed with suggested search terms that exist in the topology service.

If you select one of the suggested results, the Search Results page is displayed listing possible resource results. For each result, the name, type and other properties stored in the Elasticsearch engine are displayed.

You can expand a result in order to query the resource further and display more detailed, time-stamped information, such as its state and any associated severity levels, or when the resource was previously updated or replaced (or deleted).

You can click the **Explore Topology** button next to a result to render the topology.

If the resource that you wish to find is unique and you are confident that it is the first result in the list of search results, then instead of selecting a

result from the suggested search terms, you can choose to click the shortcut in the **Suggest** drop-down, which will render and display the topology for the closest matching resource.

Topology Search
If you conduct a Resource Search with a topology already loaded, the search functionality searches the loaded topology as well as the topology database. As you type in a search term, a drop-down list is displayed that includes suggested search results from the displayed topology listed under the **In current view** heading.

If you hover over a search result in this section, the resource is highlighted in the topology window.

If you click on a search result, the topology view zooms in on that resource and closes the search.

No. Hops
The number of relationship hops to visualize from the seed resource, with the default set at 'one'.

You define the number of relationship hops to be performed, which can be from one to four, unless this setting has been customized. See the "Defining global settings" on page 204 topic for more information on customizing the maximum hop count.

Type of Hop
The type of graph traversal used.

The two options are:

Element to Element hop type
This type performs the traversal using all element types in the graph.

Host to Host hop type
This type generates a view showing host to host connections.

Element to Host hop type
This type provides an aggregated hop view like the Host to Host type, but also includes the elements that are used to connect the hosts.

Tip: The URL captures the hopType as 'e2h'. When launching a view using a direct URL, you can use the `hopType=e2h` URL parameter.

Render
This performs the topology visualization action, rendering the topology based on the settings in the navigation toolbar.

Preemptive filtering: To prevent a large topology from being loaded, which can use considerable computational resources, you can set filters before rendering a topology.

Once rendered, the topology will refresh on a 30 second interval by default. You can pause the auto-update refresh, or select a custom interval.

Tip: The UI can time out if a large amount of data is being received. See the timeout troubleshooting section in the following topic for information on how to address this issue, if a timeout message is displayed: "Rendering (visualizing) a topology" on page 166

Additional actions > Obtain Direct URL

Open the **Additional actions** (**...**) drop-down menu, and then use the **Obtain Direct URL** option to display the Direct Topology URL dialog.

The displayed URL captures the current topology configuration, including layout type (layout orientation is not tracked).

Click **Copy** to obtain a direct-launch URL string, then click **Close** to return to the previous screen.

Use the direct-launch URL for quick access to a given topology view within DASH.

Tip: You can share this URL with all DASH users with the required permissions.

Additional actions > View System Health

Open the **Additional actions** (**...**) drop-down menu, and then use the **View System Health** option to access your Netcool Agile Service Manager deployment's system health information.

Additional actions > Edit User Preferences

Open the **Additional actions** (**...**) drop-down menu, and then use the **Edit User Preferences** option to access the User Preferences window. Click **Save**, then **Close** when done.

You can customize the following user preferences to suit your requirements:

Updates

Default auto refresh rate (seconds)

The rate at which the topology will be updated.

The default value is 30.

You must reopen the page before any changes to this user preference take effect.

Maximum number of resources to load with auto refresh enabled

When the resource limit set here is reached, auto-refresh is turned off.

The maximum value is 2000, which is also set as the default.

Tip: If you find that the default value is too high and negatively impacts your topology viewer's performance, reduce this value.

Auto render new resources

Enable this option to display new resources at the next scheduled or ad-hoc refresh as soon as they are detected.

Remove deleted topology resources

Enable this option to remove deleted resources at the next scheduled or ad-hoc refresh.

Layout

Set **Default layout type** including the layout orientation for some of the layout types. You can also configure a default layout in User Preferences.

You can choose from a number of layout types, and also set the orientation for layouts 4, 6, 7 and 8.

Tip: A change to a layout type is tracked in the URL (layout orientation is not tracked). You can manually edit your URL to change the layout type display settings.
The following numbered layout types are available:

Layout 1
A layout that simply displays all resources in a topology without applying a specific layout structure.

Layout 2
A circular layout that is useful when you want to arrange a number of entities by type in a circular pattern.

Layout 3
A grouped layout is useful when you have many linked entities, as it helps you visualize the entities to which a number of other entities are linked. This layout helps to identify groups of interconnected entities and the relationships between them.

Layout 4
A hierarchical layout that is useful for topologies that contain hierarchical structures, as it shows how key vertices relate to others with peers in the topology being aligned.

Layout 5
A force-directed (or 'peacock') layout is useful when you have many interlinked vertices, which group the other linked vertices.

Layout 6
A planar rank layout is useful when you want to view how the topology relates to a given vertex in terms of its rank, and also how vertices are layered relative to one another.

Layout 7
A rank layout is useful when you want to see how a selected vertex and the vertices immediately related to it rank relative to the remainder of the topology (up to the specified amount of hops). The root selection is automatic.

For example, vertices with high degrees of connectivity outrank lower degrees of connectivity. This layout ranks the topology automatically around the specified seed vertex.

Layout 8
A root rank layout similar to layout 7, except that it treats the selected vertex as the root. This layout is useful when you want to treat a selected vertex as the root of the tree, with others being ranked below it.

Ranks the topology using the selected vertex as the root (root selection: Selection)

Layout orientation

For layouts 4, 6, 7 and 8, you can set the following layout orientations:

- Top to bottom
- Bottom to top
- Left to right
- Right to left

Misc

Information message auto hide timeout (seconds)

The number of seconds that information messages are shown for in the UI.

The default value is 3.

Tip: If you are using a screen reader, it may be helpful to increase this value to ensure that you do not miss the message.

Screen reader support for graphical topology

You can enable the display of additional Help text on screen elements, which can improve the usability of screen readers.

You must reopen the page before any changes to this user preference take effect.

Enhanced client side logging, for problem diagnosis

If enabled, additional debug output is generated, which you can use for defect isolation.

Tip: Use this for specific defect hunting tasks, and then disable it again. If left enabled, it can reduce the topology viewer's performance.

You must reopen the page before any changes to this user preference take effect.

Visualization toolbar

The Topology Viewer visualization toolbar is displayed below the navigation toolbar, and provides you with access to functionality to manipulate the topology visualization.

Select tool submenu

When you hover over the Select tool icon, a submenu is displayed from which you can choose the **Select**, **Pan** or **Zoom Select** tool.

Select tool

Use this icon to select individual resources using a mouse click, or to select groups of resources by creating a selection area (using click-and-drag).

Pan tool

Use this icon to pan across the topology using click-and-drag on a blank area of the visualization panel.

Zoom Select tool
Use this icon to zoom in on an area of the topology using click-and-drag.

Zoom In
Use this icon to zoom in on the displayed topology.

Zoom Out
Use this icon to zoom out of the displayed topology.

Zoom Fit
Use this icon to fit the entire topology in the current view panel.

Overview Toggle
Use this icon to create the overview mini map in the bottom right corner.

The mini map provides an overview of the entire topology while you zoom in or out of the main topology. The mini map displays a red rectangle to represent the current topology view.

Layout
Use this icon to recalculate, and then render the topology layout again.

You can choose from a number of layout types and orientations.

Layout 1
A layout that simply displays all resources in a topology without applying a specific layout structure.

Layout 2
A circular layout that is useful when you want to arrange a number of entities by type in a circular pattern.

Layout 3
A grouped layout is useful when you have many linked entities, as it helps you visualize the entities to which a number of other entities are linked. This layout helps to identify groups of interconnected entities and the relationships between them.

Layout 4
A hierarchical layout that is useful for topologies that contain hierarchical structures, as it shows how key vertices relate to others with peers in the topology being aligned.

Layout 5
A peacock layout is useful when you have many interlinked vertices, which group the other linked vertices.

Layout 6
A planar rank layout is useful when you want to view how the topology relates to a given vertex in terms of its rank, and also how vertices are layered relative to one another.

Layout 7
A rank layout is useful when you want to see how a selected vertex and the vertices immediately related to it rank relative to the remainder of the topology (up to the specified amount of hops). The root selection is automatic.

For example, vertices with high degrees of connectivity outrank lower degrees of connectivity. This layout ranks the topology automatically around the specified seed vertex.

Layout 8

A root rank layout similar to layout 7, except that it treats the selected vertex as the root. This layout is useful when you want to treat a selected vertex as the root of the tree, with others being ranked below it.

Ranks the topology using the selected vertex as the root (root selection: Selection)

Layout orientation

For layouts 4, 6, 7 and 8, you can set the following layout orientations:

- Top to bottom
- Bottom to top
- Left to right
- Right to left

Filter Toolbar toggle

Use this icon to display or hide the filter toolbar. You can filter resources that are displayed in the topology, or set filters before rendering a topology.

If a filter has been applied to a displayed topology, the text 'Filtering applied' is displayed in the status bar at the bottom of the topology.

History toggle

Use this to open and close the Topology History toolbar. The topology is displayed in history mode by default.

Configure Refresh Rate

When you hover over the **Refresh Rate** icon, a submenu is displayed from which you can configure the auto-update refresh rate.

You can pause the topology data refresh, or specify the following values: 10 seconds, thirty seconds (default), one minute, or five minutes.

Resource display conventions

Deleted: A minus icon shows that a resource has been deleted since last rendered.

Displayed when a topology is updated, and in the history views.

Added: A purple plus (+) icon shows that a resource has been added since last rendered.

Displayed when a topology is updated, and in the history views.

Added (neighbors): A blue asterisk icon shows that a resource has been added using the 'get neighbors' function.

Topology visualization panel

The main panel under the visualization toolbar displays the topology.

The displayed topology consists of resource nodes and the relationship links connecting the resources. You can interact with these nodes and links using the mouse functionality.

Dragging a node

Click and drag a node to move it.

Selecting a node

Selection of a node highlights the node, and emphasizes its first-order connections by fading all other resources.

Context menu (right-click)

You open the context menu using the right-click function. The context menu provides access to the resource-specific actions you can perform.

For resource entities, you can perform the following:

Resource Details

When selected, displays a dialog that shows all the current stored properties for the specified resource in table format.

Resource Status

If statuses related to a specific resource are available, the resource will be marked with an icon depicting the status severity level, and the Resource Status option will appear in the resource context menu.

When selected, Resource Status displays a dialog that shows the time-stamped statuses related to the specified resource in table format. The Severity, Time, and State columns can be sorted, and the moment that Resource Status was selected is also time-stamped.

In addition, if any status tools have been defined, the status tool selector (three dots) is displayed next to the resource's statuses. Click the status tool selector to display a list of any status tools that have been defined, and then click the specific tool to run it. Status tools are only displayed for the states that were specified when the tools were defined.

The **state** of a status is either 'open', 'clear', or 'closed'.

The **severity** of a status ranges from 'clear' (white tick on a green square) to 'critical' (white cross on a red circle).

Table 63. Severity levels

Icon	Severity
	clear
	indeterminate
	information
	warning
	minor
	major
	critical

Comments

When selected, this displays any comments recorded against the resource.

By default, resource comments are displayed by date in ascending order. You can sort them in the following way:

- Oldest first
- Newest first
- User Id (A to Z)
- User Id (Z to A)

Users with the inasm_operator role can view comments, but not add any. Users with inasm_editor or inasm_admin roles can also add new comments. See the "Configuring DASH user roles" on page 21 topic for more information on assigning user roles.

To add a new comment, enter text into the New Comment field, and then click **Add Comment** to save.

Get Neighbors
When selected, opens a menu that displays the resource types of all the neighboring resources. Each resource type lists the number of resources of that type, as well as the maximum severity associated with each type.

You can choose to get all neighbors of the selected resource, or only the neighbors of a specific type. This lets you expand the topology in controlled, incremental steps.

Selecting **Get Neighbors** overrides any existing filters.

You can **Undo** the last neighbor request made.

Follow Relationship
When selected, opens a menu that displays all adjacent relationship types.

Each relationship type lists the number of relationships of that type, as well as the maximum severity associated with each type.

You can choose to follow all relationships, or only the neighbors of a specific type.

Show last change in timeline
When selected, will display the history timeline depicting the most recent change made to the resource.

Show first change in timeline
When selected, will display the history timeline depicting the first change made to the resource.

Recenter View
When selected, this updates the displayed topology with the specified resource as seed.

Information bar
A section at the bottom of the screen displays the current status of the rendered topology.

A timestamp on the left of the information bar indicates the time of the most recent refresh. If two time periods are being compared, both will be indicated.

Additional information on the right describes the number of resources rendered, their relationships, whether they were added or removed since the last refresh, and whether a filter has been applied.

Filter toolbar

Open and close the Filter toolbar using the **Filter** toggle in the Topology Visualization toolbar (on the left). When you have filtered your topology, click **Close** to remove the toolbar from view.

The Filter toolbar is displayed as a panel on the right-hand side of the page, and consists of a **Simple** and an **Advanced** tab. If selected, each tab provides you with access to lists of resource types and relationship types.

- **If you are filtering a topology before rendering it:** All resource types and relationship types are displayed. After rendering the topology, you can toggle the **Show all types** switch so that only types relevant to your topology are displayed.
- **If you are filtering a topology already displayed in the viewer:** Only types relevant to your topology are displayed, for example **host**, **ipaddress**, or **operatingsystem**. You you can toggle the **Show all types** switch so that all types are listed.

Simple tab

When you use the Simple tab to filter out resource or relationship types, all specified types are removed from view, including the seed resource.

It **only** removes the resources matching that type, leaving the resources below, or further out from that type, based on topology traversals.

By default, all types are **On**. Use the **Off** toggle to remove specific types from your view.

Advanced tab

The Advanced tab performs a server-side topology-based filter action.

It removes the resources matching that type, **as well as** all resources below that type.

However, the seed resource is **not** removed from view, even if it is of a type selected for removal.

Tips

Reset or invert all filters: Click **Reset** to switch all types back on, or click **Invert** to invert your selection of types filtered.

Hover to highlight: When a topology is displayed, hover over one of the filtering type options to highlight them in the topology.

Topology History toolbar

Open and close the Topology History toolbar using the **History** button in the Topology Visualization toolbar (on the left). You can hide the Topology History toolbar by clicking **Close**, which also returns the topology to update mode.

Update mode

The topology is displayed in update mode by default with Delta mode set to **Off**.

While viewing the timeline in update mode with Delta mode set to **On**, any changes to the topology history are displayed on the right hand side of the timeline, with the time pins moving apart at set intervals. By clicking **Render**, you reset the endpoint to 'now' and the pins form a single line again.

While viewing the timeline in update mode with Delta mode set to **Off**, only a single pin is displayed.

Delta mode

You toggle between delta mode **On** and **Off** using the Delta switch above the topology.

When Delta mode is **On** with Update mode also **On**, differences in topology are displayed via purple plus or minus symbols next to the affected resource.

When Delta mode is **On** with History mode **On** (that is, Update mode set to **Off**), you can compare two time points to view differences in topology.

Lock time pin

Click the **Lock** icon on a time pin's head to lock a time point in place as a reference point, and then use the second time slider to view topology changes.

History timeline

You open the Topology History toolbar using the **History** toggle in the Topology Visualization toolbar (on the left).

You use the time pins to control the topology shown. When you move the pins, the topology updates to show the topology representation at that time.

While in delta mode you can move both pins to show a comparison between the earliest pin and the latest. The timeline shows the historic changes for a single selected resource, which is indicated in the timeline title. You can lock one of the time pins in place to be a reference point.

When you first display the history timeline, coach marks (or tooltips) are displayed, which contain helpful information about the timeline functionality. You can scroll through these, or switch them off (or on again) as required.

To view the timeline for a different resource, you click on it, and the heading above the timeline changes to display the name of the selected resource. If you click on the heading, the topology centers (and zooms into) the selected resource.

The history timeline is displayed above a secondary time bar, which displays a larger time segment and indicates how much of it is depicted in the main timeline. You can use the jump buttons to move back and forth along the timeline, or jump to the current time.

You can use the time picker, which opens a calendar and clock, to move to a specific second in time.

To view changes made during a specific time period, use the two time sliders to set the time period. You can zoom in and out to increase or decrease the granularity using the + and - buttons on the right, or by double-clicking within a time frame. The most granular level you can display is an interval of one second. The granularity is depicted with time indicators and parallel bars, which form 'buckets' that contain the recorded resource change event details.

The timeline displays changes to a resource's state, properties, and its relationships with other resources. These changes are displayed through

color-coded bars and dash lines, and are elaborated on in a tooltip displayed when you hover over the change. You can exclude one or more of these from display.

Resource state changes
The timeline displays the number of state changes a resource has undergone.

Resource property changes
The timeline displays the number of times that resource properties were changed.

Each time that property changes were made is displayed as one property change event regardless of whether one or more properties were changed at the time.

Resource relationship changes
The number of relationships with neighboring resources are displayed, and whether these were changed.

The timeline displays when relationships with other resources were changed, and also whether these changes were the removal or addition of a relationship, or the modification of an existing relationship.

Update manager

If auto-updates have been turned off, the Update Manager informs you if new resources have been detected. It allows you to continue working with your current topology until you are ready to integrate the new resources into the view.

The Update Manager is displayed in the bottom right of the screen.

The Update Manager provides you with the following options:

Show details
Displays additional resource information.

Render
Integrates the new resources into the topology.

Choosing this option will recalculate the topology layout based on your current display settings, and may therefore adjust the displayed topology significantly.

Cogwheel icon
When clicked, provides you with quick access to change your user preferences:

- **Enable auto-refresh:** Switches auto-refresh back on, and disables the Update Manager.
- **Remove deleted resources:** Removes the deleted resources from your topology view when the next topology update occurs.

Hide Reduces the Update Manager to a small purple icon that does not obstruct your current topology view.

When you are ready to deal with the new resources, click on the icon to display the Update Manager again.

Topology tools reference

This reference topic describes the Netcool Agile Service Manager Topology tools functionality.

Topology Tools - Details

The Topology Tools - Details page is displayed when you select a right-click tool to edit it, or when you create a new tool. Here you define a tool's name and label as a minimum.

Name Unique name used as an internal reference.

Required.

Menu label
The menu label is the text displayed in the context menu.

This can be the same name as used by other tools, which is why the unique name is required.

Required

Description
A description to help administrator users record the tool's purpose.

Not displayed in the context menu.

Optional.

Menu priority
The menu priority slider defines where in the context menu the tool is displayed.

For example, tools with a priority of two will be displayed higher in the menu than tools that have a priority of four.

Available values are one to ten.

Optional.

Navigation
You can move to the next page by using the page selector.

The minimum requirement to save the tool and open the Topology Tools - Implementation page is the name and label.

Topology Tools - Implementation

The Topology Tools - Implementation page is displayed after you have completed the Topology Tools - Details page. Here you define the tool using valid JavaScript. To help you create tools, you have access to the following custom helper functions:

asmProperties
The tool implementation has access to the properties of the relevant **resource**, **relationship** or **status** via the asmProperties JavaScript object, which contains all the properties.

You can access the properties using standard JavaScript, but you must protect against a value not being present.

For example if you intend to use the property 'latitude', you must verify that it is present before using it. To do so, use the following check command:

```
asmProperties.hasOwnProperty('latitude')
```

If the property is present, the Boolean value `true` will be returned.

Status tools properties

When creating **status** tools, you use JavaScript that is similar to the script that you use when creating **resource** or **relationship** tools. However, the properties you use in your status tool scripts, such as asmProperties, reference the properties for the **status** item; unlike the properties you use in your resource or relationship tool scripts, which reference the properties for the resources or relationships. For example, if you use asmProperties.location in a status tool script, there must be a corresponding 'location' property in the status record.

When creating status tools, the asmProperties object has a property that takes the form of an array called **`resources`**, which represents the resources in the topology with which this status is associated. Each item in the resources array is an object with properties that represent the properties of that resource. For example, if a status is associated with two resources, the **`uniqueId`** property of the first of those two resources could be referenced in the script by using `asmProperties.resources[0].uniqueId`

In addition, you can access the properties of a resource against which you are running a status tool by using the **`asmSourceProperties`** object when scripting the status tool.

asmSourceProperties

You can access information about the source properties of any **relationships** or **status** the custom tool is acting on via the asmSourceProperties JavaScript object.

Example of using the source resource properties in a custom relationship stroke definition:

```
if (asmSourceProperties.myProp === 'high') {
    return 'blue';
} else {
    return 'black';
}
```

Remember: The arrows indicating a relationship point from the source to the target.

asmTargetProperties

You can access information about the target properties of **relationships** the custom tool is acting on via the asmTargetProperties JavaScript object.

asmFunctions

You can use a number of other helper functions, which are accessed from the asmFunctions object, which includes the following:

showConfirmationPopup(title, message, onOk)

Creates a popup confirmation allowing the tool to confirm an action.

Takes a title and message, which is displayed on the popup, and a function definition, which is run if the user clicks the OK button on the popup.

showToasterMessage(status, message)
Shows a popup toaster with the appropriate status coloring and message.

showPopup(title, text)
Shows a popup with a given title and text body, which can be generated based on the properties of the resource or relationship.

showIframe(url)
Displays a popup filling most of the page which wraps an iframe showing the page of the given URL.

Allows you to embed additional pages.

sendPortletEvent(event)
Allows you to send DASH portlet events from the Topology Viewer that can be used to manipulate other DASH portlets, such as the Event Viewer within IBM Tivoli Netcool/OMNIbus Web GUI.

Note: You can send events to other DASH portlets only if you are running Agile Service Manager within DASH (rather than in a direct-launch browser window), and if the receiving DASH portlets subscribe to the types of events being sent. See the "sendPortletEvent examples" on page 197 topic for more information.

asmFunctions.getResourceStatus(<resource_id>, <callback_function>, [<time_stamp>])
Allows you to request status information from a tool definition for a given resource using its **_id** parameter.

resource_id
Required

Can be obtained from a resource via `asmProperties._id` and from a relationship using `asmSourceProperties._id` or `asmTargetProperties._id`

callback_function
Required

Is called once the status data has been collected from the topology service, with a single argument containing an array of status objects

time_stamp
Optional

Unix millisecond timestamp to get the status from a given point in history

The following example prints the status information of a source resource from a relationship context to the browser console log:

```
let printStatusCallback = function(statuses) {
    statuses.forEach(function(status) {
        console.log('status:', status.status,
                    'state:', status.state,
                    'severity:', status.severity,
                    'time:', new Date(status.time));
```

```
    })
}
asmFunctions.getResourceStatus(asmSourceProperties._id,
printStatusCallback);
```

Topology Tools - Conditions

The Topology Tools - Conditions page is displayed after you have completed the Topology Tools - Implementation page. Here you select the resource, relationship or status that will display the tool in their context menus.

Applicable item type for tool definition
From this drop-down, select the types to which the tool is applicable: **Resource**, **Relationship**, **Resource and Relationship**, or **Status**.

Depending on your selection, a number of check boxes are displayed, which you use to configure which resources, relationships or states are included.

All types / All states
Select this option if you want the tool to be displayed for all resource and relationship types, or all states (for Status).

The tool will also be displayed for any specific types not listed here.

Resource types
Select one or more resource types from the list displayed.

Relationship types
Select one or more relationship types from the list displayed.

Status Select from the following possible states for which the tool will be available:

- **Open**
- **Clear**
- **Closed**

Remember: When creating status tools, the properties you use in your status tool scripts reference the properties for the status item, while the properties you use in your resource or relationship tools reference the properties for the resources or relationships.

Custom icons reference

This reference topic describes the Netcool Agile Service Manager Custom Icons functionality.

Custom Icons

The Custom Icons page is displayed when you select **Administration** from the DASH menu, and then click **Custom Icons** under the Agile Service Management heading.

The Custom Icons page displays the following buttons.

New Opens the Configure Custom Icon page

'Refresh' symbol
Reloads the icon information from the topology service

In addition, the Custom Icons page displays the following icon information in table format.

Name Unique icon name

Icon The icon itself

If you hover over a custom icon, it will be enlarged and displayed inside a circle to show what it will look like within a topology view.

Last Updated
Date and timestamp

Size (KB)
Size of the icon SVG in KB

'Edit' symbol
Opens the Configure Custom Icon page

'Bin' symbol
Deletes an icon.

If assigned to a resource type, a warning is displayed.

Category
Sorts icons by category

Configure Custom Icon

The Configure Custom Icon page is displayed when you select an icon on the Custom Icons page to edit it, or when you create a new icon. Here you define an icon's name and SVG XML (both required) using the provided SVG XML editor.

Name Each icon must have a name, which uniquely identifies the icon when assigning it to a type.

You cannot change the name of an existing icon. If you want an icon to have a different name, create a new icon, then delete the old one.

SVG XML
Use the XML editor to enter or edit the SVG text.

Each icon definition must be valid svg xml with a given viewBox, which is important to ensure scaling of the image. The SVG editor rejects any invalid XML entered.

The svg definition must include inline styling of the image, such as stroke color and fill color. If style classes are used, naming must be unique for each svg image to prevent class definitions from being overwritten.

The XML editor includes a Preview area where the results of your SVG edits are displayed.

Category
Optionally, each icon can be assigned to a category. You can use categories to group icons of the same type or function together.

If you sort the full list of icons by Category, icons with the same category are displayed together.

Example: Use the following definition for the 'disk' icon as guidance:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64">
   <ellipse style="fill-opacity:0;stroke:currentColor;stroke-width:12.12270069;"
id="path4139" cx="33.627117" cy="32.949142" rx="16.803904" ry="17.210684"/>
   <circle cx="33.827423" cy="33.055576" r="3.3037829"/>
</svg>
```

Example sysctl.conf file

The following example of a `sysctl.conf` file shows settings that have been used in testing.

/etc/sysctl.conf

Tip: Optimize Cassandra and ElasticSearch Kernel parameters by either disabling Swap, or setting the Kernel **`vm.swappiness`** parameter to 1.

To customize your sysctl.conf file, first back-up the original file, then edit it, before restarting the system. The default location is `/etc/sysctl.conf`

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.

vm.swappiness = 1
vm.dirty_background_ratio = 3
vm.dirty_ratio = 80
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100

kernel.shmmax = 4398046511104
kernel.shmall = 1073741824

kernel.sem = 250 256000 100 16384

net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576

fs.aio-max-nr = 1048576

kernel.panic_on_oops = 1

fs.file-max = 6815744

net.ipv4.tcp_tw_recycle = 1

net.ipv4.tcp_tw_reuse = 1

net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.tcp_syncookies = 1

net.core.somaxconn = 1024

kernel.shmmni = 16384

net.ipv4.ip_local_port_range = 9000 65535

kernel.msgmnb = 65536

kernel.msgmax = 65536

kernel.shmmax = 540971653120

kernel.shmall = 4294967296
```

Swagger reference

Specific links to Agile Service Manager Swagger documentation are included in many of the topics, as and when useful. This topic summarizes some of that information in a single location, for example by listing the default ports and Swagger URLs for each Agile Service Manager service.

Swagger overview

Swagger is an open source software framework, which includes support for automated documentation and code generation. You can find more information on the Swagger website: https://swagger.io/docs/

Swagger
: Agile Service Manager uses Swagger for automated documentation generation and utilizes a Swagger server for each micro-service.

  You can access and explore the REST APIs of the topology service and observers using Swagger via the proxy service.

For example

- To access the **Topology Service** via Swagger, use the following URL: `https://<your host>/1.0/topology/swagger`
- To access the **Event Observer** service via Swagger, use the following URL: `https://<your host>/1.0/event-observer/swagger`

Important:

For the on-prem version of Agile Service Manager, you access the micro-services through the proxy service (nasm-nginx), which requires a proxy user and password to be configured for Nginx.

The default values for the user name and password are:

username
: `asm`

password
: `asm`

Default Swagger URLs

The following tables show the default Swagger URLs for Agile Service Manager services and observers.

Table 64. Default Swagger URLs for Agile Service Manager services

Service	Swagger URL	ICP
layout	`https://<your host>/1.0/layout/swagger`	yes
merge	`https://<your host>/1.0/merge/swagger`	yes
search	`https://<your host>/1.0/search/swagger`	yes
topology	`https://<your host>/1.0/topology/swagger`	yes

Table 65. Default Swagger URLs for Agile Service Manager observers

Observer	Swagger URL	ICP
alm-observer	`https://<your host>/1.0/alm-observer/swagger`	no
aws-observer	`https://<your host>/1.0/aws-observer/swagger`	yes
bigfixinventory-observer	`https://<your host>/1.0/bigfixinventory-observer/swagger`	yes
cienablueplanet-observer	`https://<your host>/1.0/cienablueplanet-observer/swagger`	yes
ciscoaci-observer	`https://<your host>/1.0/ciscoaci-observer/swagger`	yes
contrail-observer	`https://<your host>/1.0/contrail-observer/swagger`	yes
dns-observer	`https://<your host>/1.0/dns-observer/swagger`	yes
docker-observer	`https://<your host>/1.0/docker-observer/swagger`	yes
dynatrace-observer	`https://<your host>/1.0/dynatrace-observer/swagger`	yes
event-observer	`https://<your host>/1.0/event-observer/swagger`	yes
file-observer	`https://<your host>/1.0/file-observer/swagger`	yes
ibmcloud-observer	`https://<your host>/1.0/ibmcloud-observer/swagger`	yes
itnm-observer	`https://<your host>/1.0/itnm-observer/swagger`	yes
kubernetes-observer	`https://<your host>/1.0/kubernetes-observer/swagger`	yes
newrelic-observer	`https://<your host>/1.0/newrelic-observer/swagger`	yes
openstack-observer	`https://<your host>/1.0/openstack-observer/swagger`	yes
rest-observer	`https://<your host>/1.0/rest-observer/swagger`	yes
servicenow-observer	`https://<your host>/1.0/servicenow-observer/swagger`	yes
taddm-observer	`https://<your host>/1.0/taddm-observer/swagger`	yes
vmvcenter-observer	`https://<your host>/1.0/vmvcenter-observer/swagger`	yes
vmwarensx-observer	`https://<your host>/1.0/vmwarensx-observer/swagger`	yes
zabbix-observer	`https://<your host>/1.0/zabbix-observer/swagger`	yes

Sizing reference

The default deployment configuration will start three instances of Cassandra, Elasticsearch, Kafka and Zookeeper. This topic lists the compute and storage resources required for a default production deployment (size1). To ensure resiliency, you need a minimum of three worker nodes in your cluster with this configuration.

Compute resources

Table 66. Compute resources for a size0 deployment. This table summarizes the required compute resources for all components of a default production (size0) deployment.

Service	CPU requests	CPU limits	Memory requests	Memory limits
alm-observer	0.2	0.8	200Mi	450Mi
aws-observer	0.2	0.8	200Mi	450Mi
bigfixinventory-observer	0.2	0.8	200Mi	450Mi
cienablueplanet-observer	0.2	0.8	200Mi	450Mi
ciscoaci-observer	0.2	0.8	200Mi	450Mi
contrail-observer	0.2	0.8	200Mi	450Mi
docker-observer	0.2	0.8	200Mi	450Mi
dns-observer	0.2	0.8	200Mi	450Mi
dynatrace-observer	0.2	0.8	200Mi	450Mi
docker-observer	0.2	0.8	200Mi	450Mi
event-observer	0.2	0.8	200Mi	450Mi
file-observer	0.2	0.8	200Mi	450Mi
ibmcloud-observer	0.2	0.8	200Mi	450Mi
itnm-observer	0.2	0.8	200Mi	450Mi
kubernetes-observer	0.2	0.8	200Mi	450Mi
newrelic-observer	0.2	0.8	200Mi	450Mi
openstack-observer	0.2	0.8	200Mi	450Mi
rest-observer	0.2	0.8	200Mi	450Mi
servicenow-observer	0.2	0.8	200Mi	450Mi
taddm-observer	0.2	0.8	200Mi	450Mi
vmvcenter-observer	0.2	0.8	200Mi	450Mi
vmwarensx-observer	0.2	0.8	200Mi	450Mi
zabbix-observer	0.2	0.8	200Mi	450Mi
layout	0.4	0.8	450Mi	1050Mi

Table 66. Compute resources for a size0 deployment (continued). This table summarizes the required compute resources for all components of a default production (size0) deployment.

Service	CPU requests	CPU limits	Memory requests	Memory limits
merge	0.4	0.8	450Mi	550Mi
search	0.4	0.8	450Mi	550Mi
topology	2.5	4.0	450Mi	700Mi
elasticsearch	0.2	1.0	1200Mi	2800Mi
ui-api	0.2	0.8	200Mi	450Mi
kafka	0.2	1.0	600Mi	800Mi
kafkarest	0.2	1.0	350Mi	600Mi
zookeeper	0.1	0.5	350Mi	450Mi
cassandra	1.0	4.0	6Gi	6Gi

Table 67. Compute resources for a size1 deployment. This table summarizes the required compute resources for a minimal (size1) deployment of all Agile Service Manager components, including observers. A size0 deployment is suitable for a test or proof-of-concept deployment only.

Service	CPU requests	CPU limits	Memory requests	Memory limits
alm-observer	0.5	1.0	350Mi	750Mi
aws-observer	0.5	1.0	350Mi	750Mi
bigfixinventory-observer	0.5	1.0	350Mi	750Mi
cienablueplanet-observer	0.5	1.0	350Mi	750Mi
ciscoaci-observer	0.5	1.0	350Mi	750Mi
contrail-observer	0.5	1.0	350Mi	750Mi
docker-observer	0.5	1.0	350Mi	750Mi
dns-observer	0.5	1.0	350Mi	750Mi
dynatrace-observer	0.5	1.0	350Mi	750Mi
docker-observer	0.5	1.0	350Mi	750Mi
event-observer	0.5	1.0	350Mi	750Mi
file-observer	0.5	1.0	350Mi	750Mi
ibmcloud-observer	0.5	1.0	350Mi	750Mi
itnm-observer	0.5	1.0	350Mi	750Mi
kubernetes-observer	0.5	1.0	350Mi	750Mi
newrelic-observer	0.5	1.0	350Mi	750Mi
openstack-observer	0.5	1.0	350Mi	750Mi
rest-observer	0.5	1.0	350Mi	750Mi

Table 67. Compute resources for a size1 deployment (continued). This table summarizes the required compute resources for a minimal (size1) deployment of all Agile Service Manager components, including observers. A size0 deployment is suitable for a test or proof-of-concept deployment only.

Service	CPU requests	CPU limits	Memory requests	Memory limits
servicenow-observer	0.5	1.0	350Mi	750Mi
taddm-observer	0.5	1.0	350Mi	750Mi
vmvcenter-observer	0.5	1.0	350Mi	750Mi
vmwarensx-observer	0.5	1.0	350Mi	750Mi
zabbix-observer	0.5	1.0	350Mi	750Mi
layout	2.0	4.0	700Mi	2500Mi
merge	1.0	1.5	1250Mi	1500Mi
search	1.0	1.5	600Mi	800Mi
topology	3.0	5.0	1200Mi	3600Mi
elasticsearch	1.0	2.5	2400Mi	4000Mi
ui-api	0.5	1.0	350Mi	750Mi
kafka	0.5	1.5	1200Mi	1600Mi
kafkarest	0.2	1.0	350Mi	600Mi
zookeeper	0.2	1.0	350Mi	450Mi
cassandra	4.0	6.0	16Gi	16Gi

Storage

Table 68. Storage requirements for a size1 deployment. This table summarizes the storage requirements for a default production deployment, which equates to approximately 150 GB per worker node.

Service	Storage (Gi)
cassandra-0	50
cassandra-1	50
cassandra-2	50
elasticsearch-0	75
elasticsearch-1	75
elasticsearch-2	75
kafka-0	15
kafka-1	15
kafka-2	15
zookeeper-0	5
zookeeper-1	5
zookeeper-2	5

Installation parameters

This topics lists the installation parameters you can override during a Helm installation.

Configurable Helm installation parameters

You override the Helm installation parameters by adding them to the Helm install command as follows:

```
--set key=value[,key=value]
```

Table 69. Helm installation parameters

Parameter	Description	Default
asm.almObserver.enabled	Option to install the Agile Lifecycle Manager observer	false
asm.awsObserver.enabled	Option to install the Amazon Web Services observer	false
asm.bigfixinventoryObserver.enabled	Option to install the BigFix Inventory observer	false
asm.cienablueplanetObserver.enabled	Option to install the Ciena Blue Planet observer	false
asm.ciscoaciObserver.enabled	Option to install the Cisco ACI observer	false
asm.contrailObserver.enabled	Option to install the Juniper Contrail observer	false
asm.dnsObserver.enabled	Option to install the DNS observer	false
asm.dockerObserver.enabled	Option to install the Docker observer	false
asm.dynatraceObserver.enabled	Option to install the Dynatrace observer	false
asm.fileObserver.enabled	Option to install the File observer	false
asm.ibmcloudObserver.enabled	Option to install the IBM Cloud observer	false
asm.itnmObserver.enabled	Option to install the ITNM observer	false
asm.newrelicObserver.enabled	Option to install the New Relic observer	false
asm.openstackObserver.enabled	Option to install the OpenStack observer	false
asm.restObserver.enabled	Option to install the REST observer	false
asm.servicenowObserver.enabled	Option to install the ServiceNow observer	false
asm.taddmObserver.enabled	Option to install the TADDM observer	false
asm.vmvcenterObserver.enabled	Option to install the VMware vCenter observer	false

Table 69. Helm installation parameters (continued)

Parameter	Description	Default
asm.vmwarensxObserver.enabled	Option to install the VMware NSX observer	false
asm.zabbixObserver.enabled	Option to install the Zabbix observer	false
license	Have you read and agree to the License agreement? set to 'accept'	not-accepted
noi.releaseName	The name of the Helm release of NOI to connect to.	noi
global.image.repository	Docker registry to pull ASM images from	
global.ingress.api.enabled	Option to enable the creation of ingress objects for the application endpoints	true
global.ingress.domain	Optional hostname to bind to the ingress rules, which must resolve to an proxy node. Multiple deployments of this chart will need to specify different values.	
global.ingress.tlsSecret	Optional TLS secret for the ingress hostname.	
global.persistence.enabled	Option to disable the requests for PersistentVolumes, for test and demo only.	true
global.persistence.storageSize.cassandradata	Option to configure the requested amount of storage for Cassandra	50Gi
global.persistence.storageSize.kafkadata	Option to configure the requested amount of storage for Kafka	15Gi
global.persistence.storageSize.zookeeperdata	Option to configure the requested amount of storage for Zookeeper	5Gi
global.persistence.storageSize.elasticdata	Option to configure the requested amount of storage for Elasticsearch	75Gi
global.cassandraNodeReplicas	The number of instances to run for Cassandra	3
global.elasticsearch.replicaCount	The number of instances to run for Elasticsearch	3
global.environmentSize	'size0' requests fewer resources and is suitable for test and demo. Choose 'size1' for a production deployment.	size1

Table 69. Helm installation parameters (continued)

Parameter	Description	Default
global.kafka.clusterSize	The number of instances to run for Kafka	3
global.zookeeper.clusterSize	The number of instances to run for Zookeeper	3

Notices

This information applies to the PDF documentation set for IBM® Netcool® Agile Service Manager.

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 give 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. 1623-14, Shimotsuruma, Yamato-shi Kanagawa 242-8502 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 might 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.

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 JBF1/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 measurement 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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, 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 Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds 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 product and service names might be trademarks of IBM or other companies.

Printed in USA