HP Universal CMDB

Software Version: Content Pack 15.00 (CP15)

Discovery and Integrations Content Guide - Discovery Modules

Document Release Date: January 2015
Software Release Date: January 2015

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hp.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support site at: https://softwaresupport.hp.com.

This website provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hp.com/web/softwaresupport/access-levels.

HP Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this website is http://h20230.www2.hp.com/sc/solutions/index.jsp.

Contents

Part 1: Accurate Dependency Mapping

Chapter 1: Database

This chapter includes:

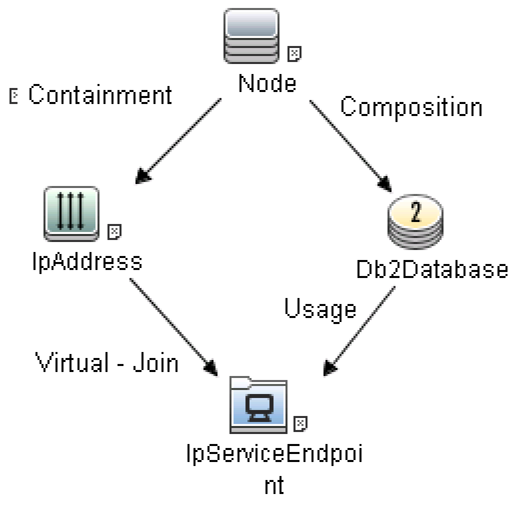
DB2 Dependencies Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and DB2 Database.

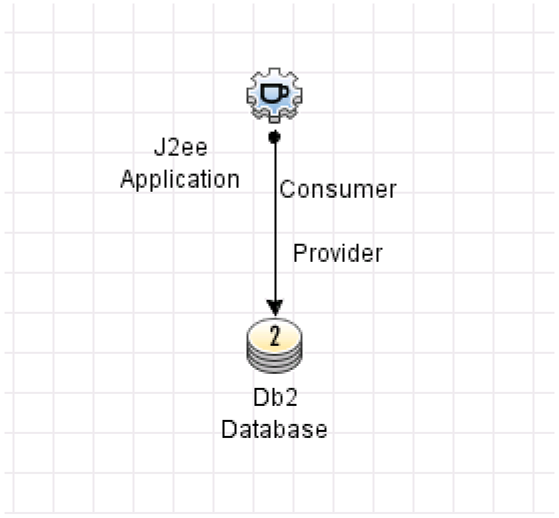
Trigger TQL



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	None
Db2Database	None

Topology Map

The **DB2 Dependencies** topology is shown below.



Supported Policy

This job does not support DB2 cluster and mainframe.

Adapter Information

This job uses the **DB2 Dependencies** adapter.

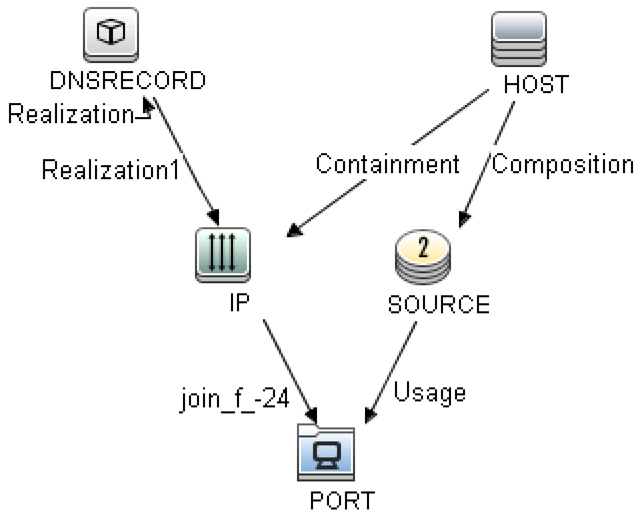
Adapter Type

Work Flow adapter

Input CIT

Db2Database

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
DATABASE_NAME	${SOURCE.name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PORT	${PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	AccurateDependencyMapping.py	Mandatory

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

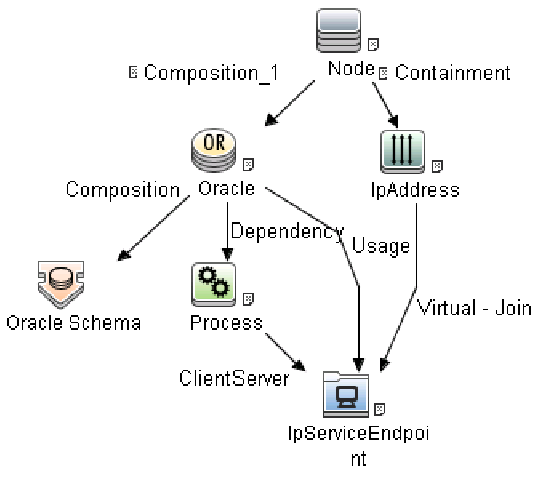
Oracle Dependencies Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and Oracle Instance.

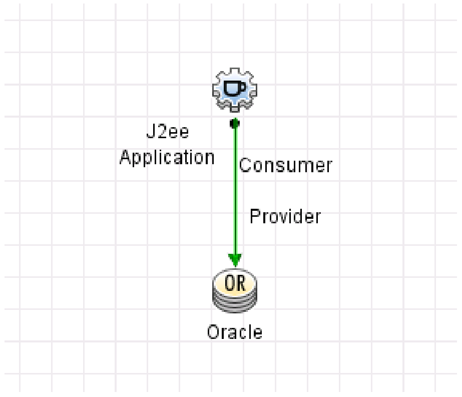
Trigger TQL



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	None
Oracle	Oracle has no Oracle Schema with Composition relationship
Oracle Schema	None
Process	None

Topology Map

The **Oracle Dependencies** topology is shown below.



Supported Policy

This job supports:

- Both Oracle single instance and RAC
- Both type 2 (thick) and type 4 (thin) driver

Adapter Information

This job uses the **Oracle Dependencies** adapter.

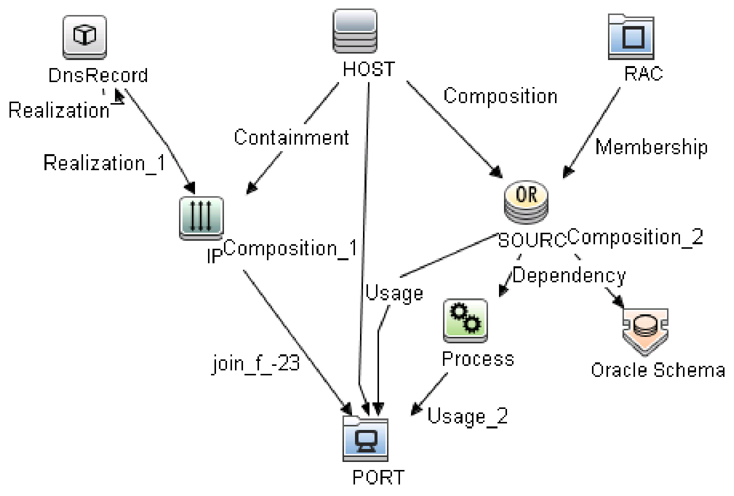
Adapter Type

Work Flow adapter

Input CIT

Oracle

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
DATABASE_NAME	${SOURCE.name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PORT	${PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}
SERVICE_NAME	${RAC.rac_servicename:}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	AccurateDependencyMapping.py	Mandatory

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

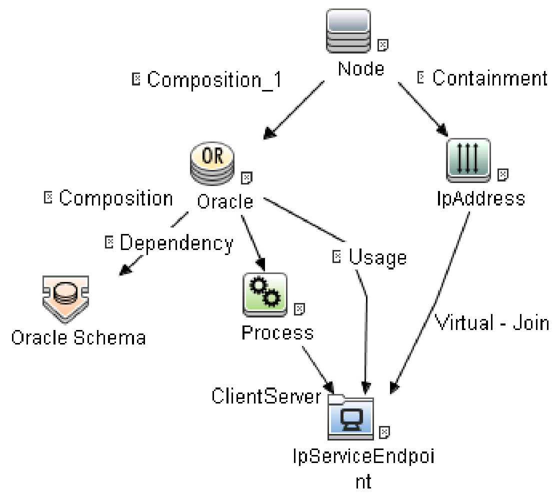
Oracle Schema Dependencies Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and Oracle Schema.

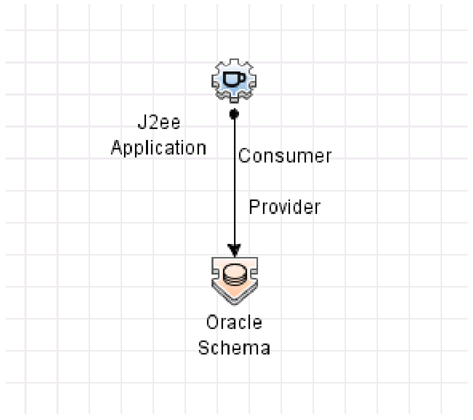
Trigger TQL



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	None
Oracle	Oracle must link to Oracle Schema with Composition relationship
Oracle Schema	None
Process	None

Topology Map

The **Oracle Schema Dependencies** topology is shown below.



Supported Policy

This job supports:

- Both Oracle single instance and RAC
- Both type 2 (thick) and type 4 (thin) driver

Adapter Information

This job uses the **Oracle Schema Dependencies** adapter.

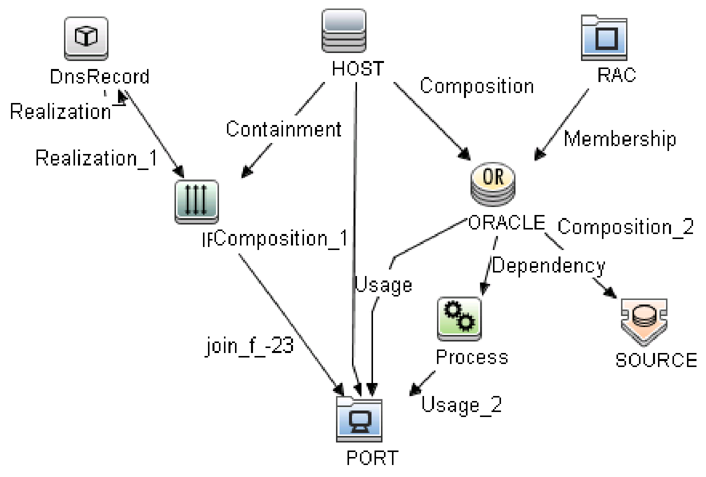
Adapter Type

Work Flow adapter

Input CIT

Oracle Schema

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PORT	${PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}
SCHEMA_NAME	${SOURCE.name}
SERVICE_NAME	${RAC.rac_servicename:}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

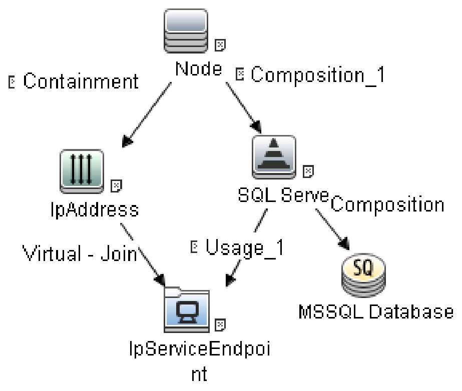
SQL Server Dependencies Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and MSSQL Database.

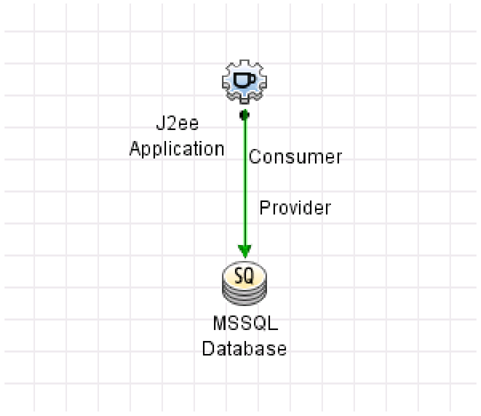
Trigger TQL



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	None
SQL Server	None
MSSQL Database	None

Topology Map

The **SQL Server Dependencies** topology is shown below.



Supported Policy

This job supports SQL Server single instance and cluster.

Adapter Information

This job uses the **SQL Server Dependencies** adapter.

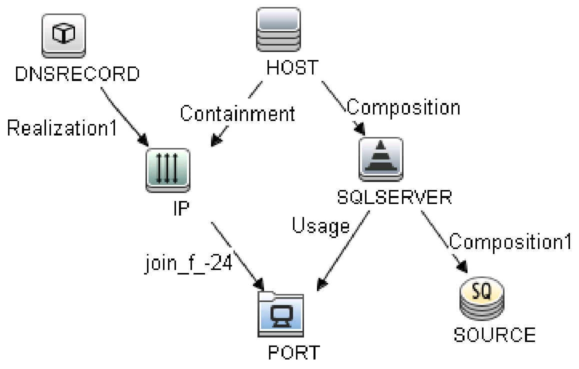
Adapter Type

Work Flow adapter

Input CIT

MSSQL Database

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
DATABASE_NAME	${SOURCE.name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PORT	${PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	AccurateDependencyMapping.py	Mandatory

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

Chapter 2: J2EE Applications

This chapter includes:

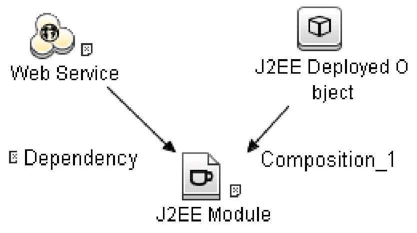
J2EE Application Dependencies via WebService Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplications via WebService.

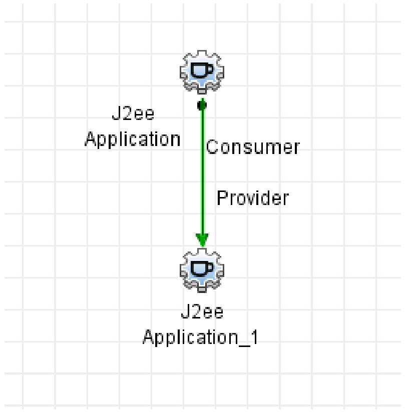
Trigger TQL



Node Name	Condition
J2EE Deployed Object	None
J2EE Module	None
Web Service	None

Topology Map

The **J2EE Application Dependencies via WebService** topology is shown below.



Supported Policy

This job supports:

- WebService invocation
- Two J2ee Applications running in one WebSphere cell
- Two J2ee Applications running in different WebSphere cells

Adapter Information

This job uses the **J2EE Application Dependencies via WebService** adapter.

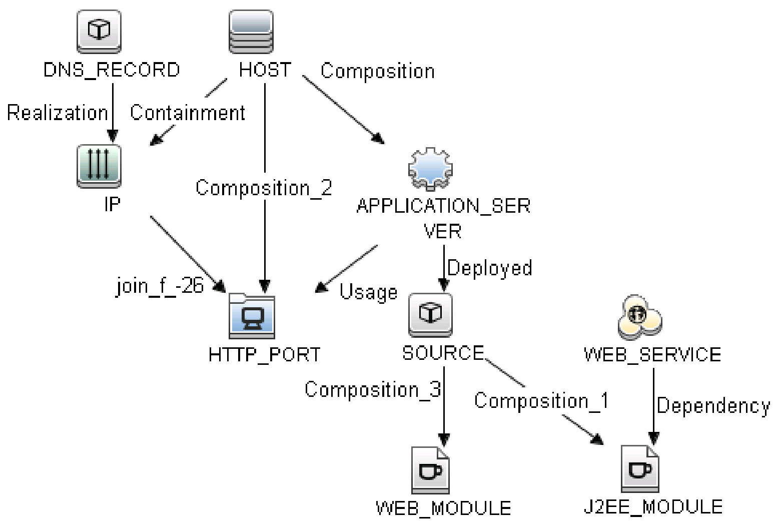
Adapter Type

Work Flow adapter

Input CIT

J2eeApplication

Input TQL



Triggered CI Data

Name	Value
ALTER_DNS_NAME	${IP.authoritative_dns_name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${APPLICATION_SERVER.application_ip}
PORT	${HTTP_PORT.network_port_number:}
PROVIDERTYPE	${SOURCE.root_class}
CONTEXT_ROOT	${WEB_MODULE.j2eemanagedobject_contextroot:}
WEBSERVICE_NAME	${WEB_SERVICE.name}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

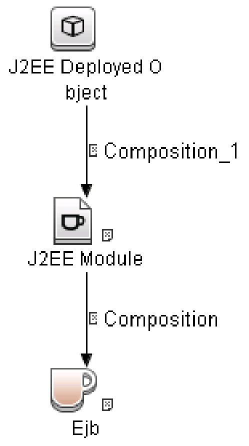
J2EE Application Dependencies via JNDI Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplications via JNDI.

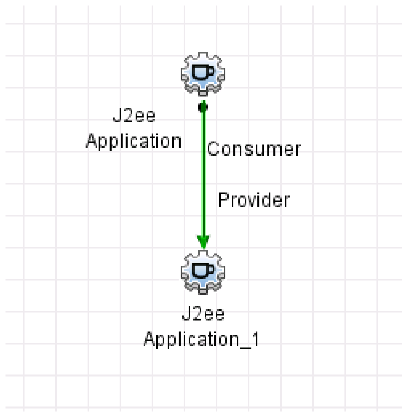
Trigger TQL



Node Name	Condition
J2EE Deployed Object	None
J2EE Module	None
Ejb	None

Topology Map

The **J2EE Application Dependencies via JNDI** topology is shown below.



Supported Policy

This job supports:

- EJB 2.0 invocation
- Two J2ee Applications running in one WebSphere cell
- Two J2ee Applications running in different WebSphere cells

Note: EJB 3.x annotation is not supported.

Adapter Information

This job uses the **J2EE Application Dependencies via JNDI** adapter.

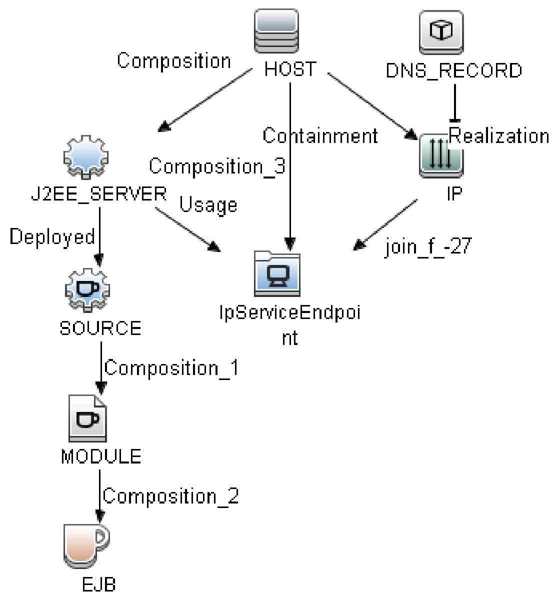
Adapter Type

Work Flow adapter

Input CIT

J2eeApplication

Input TQL



Triggered CI Data

Name	Value
ALTER_DNS_NAME	${IP.authoritative_dns_name:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PROVIDERTYPE	${SOURCE.root_class}
EJB_JNDI_NAME	${EJB.j2eemanagedobject_jndiname:}
EJB_NAME	${EJB.name}
EJB_NAME_IN_NAMESPACE	${EJB.ejb_nameinnamespace:}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

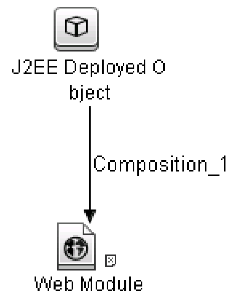
J2EE Application Dependencies via Context Root Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and BusinessElement via Context Root.

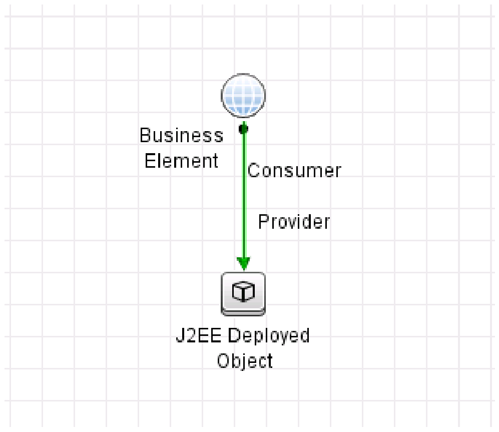
Trigger TQL



Node Name	Condition
J2EE Deployed Object	None
Web Module	None

Topology Map

The **J2EE Application Dependencies via Context Root** topology is shown below.



Adapter Information

This job uses the **J2EE Application Dependencies via Context Root** adapter.

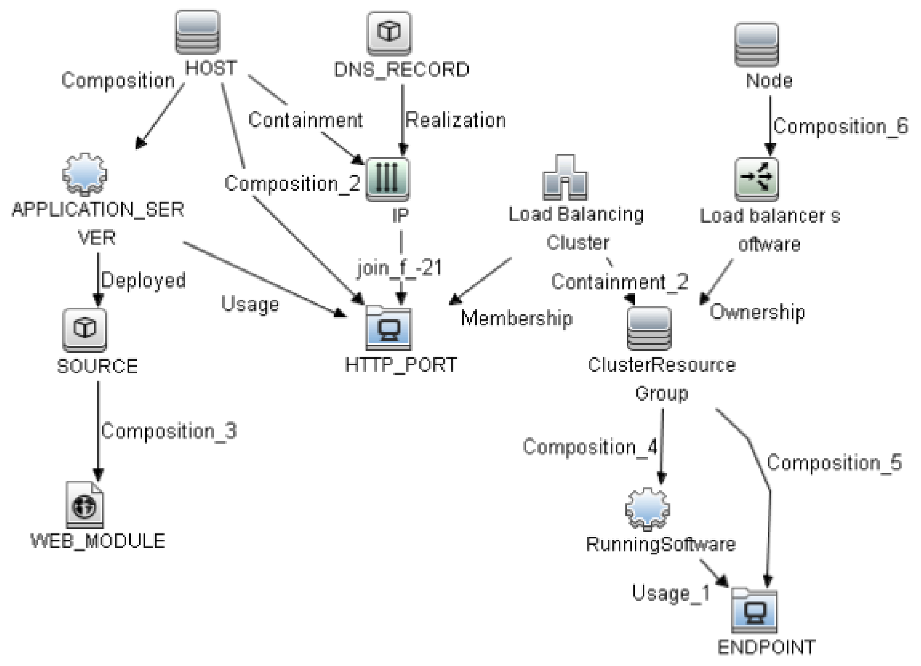
Adapter Type

Work Flow adapter

Input CIT

J2EE Deployed Object

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
CONTEXT_ROOT	${WEB_MODULE.j2eemanagedobject_contextroot:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${APPLICATION_SERVER.application_ip}
PORT	${HTTP_PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}
VIRTUAL_IP_ADDRESS	${ENDPOINT.bound_to_ip_address:}
VIRTUAL_PORT	${ENDPOINT.network_port_number:}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

Chapter 3: Messaging Servers

This chapter includes:

JMS Destination Dependencies via JNDI Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between J2eeApplication and JMS Destination via JNDI.

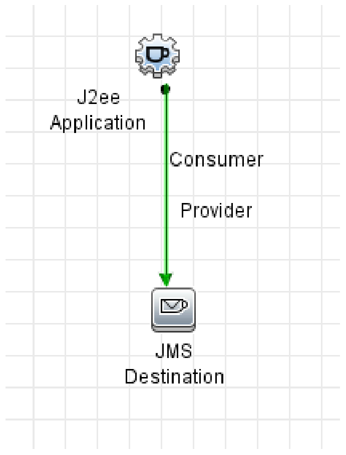
Trigger TQL

JMS Destination

Node Name	Condition
JMS Destination	None

Topology Map

The **JMS Destination Dependencies via JNDI** topology is shown below.



Adapter Information

This job uses the **JMS Destination Dependencies via JNDI** adapter.

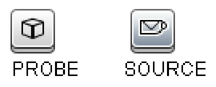
Adapter Type

Work Flow adapter

Input CIT

JMS Destination

Input TQL



Triggered CI Data

Name	Value
JMS_NAME	${SOURCE.name}
JNDI_NAME	${SOURCE.j2eemanagedobject_jndiname:}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	AccurateDependencyMapping.py	Mandatory

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

MessageQueue Dependencies via JNDI Job

This section includes details about the job.

Introduction

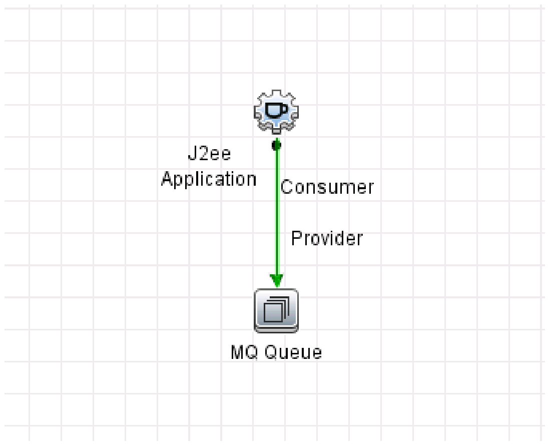
This job discovers and creates consumer-provider linkages between J2eeApplication and MQ Queue via JNDI.

Trigger TQL

Node Name	Condition
MQQueue	None

Topology Map

The **MessageQueue Dependencies via JNDI** topology is shown below.



Supported Policy

This job supports:

- WebSphere default messaging provider
- WebSphere MQ JMS provider

Adapter Information

This job uses the **MessageQueue Dependencies via JNDI** adapter.

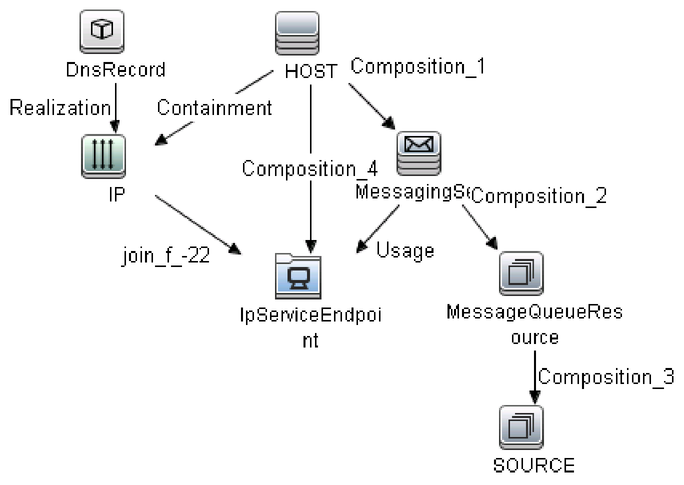
Adapter Type

Work Flow adapter

Input CIT

MQ Queue

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
DNS_NAME	${DnsRecord.name:}
IP_ADDRESS	${IP.name:}
MQ_NAME	${SOURCE.name}
PORT	${IpServiceEndpoint.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

Chapter 4: Running Software

This chapter includes:

Running Software Dependencies via TCP Connection Job

This section includes details about the job.

Introduction

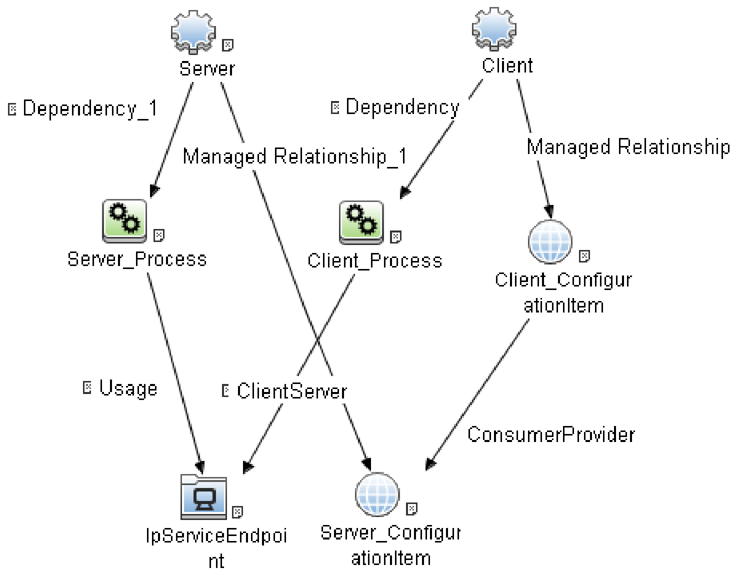
This job discovers and creates consumer-provider linkages between two running software only in the following situations:

- A client-server linkage has already been created between the Client Process and the Server IpServiceEndpoint by the **Next-Hop Provider** job with TCP connections.

  Note: If the **Next-Hop Provider** job cannot find the next hop via configuration files, it will try to find the next hop by TCP connections.

- No consumer-provider linkage exists between configuration items under the client running software and server running software.

Trigger TQL



Node Name	**Condition**
Client	Client's ConfigurationItems have no consumer-provider linkage with Server's ConfigurationItems
Server	Server's ConfigurationItems have no consumer-provider linkage with Client's ConfigurationItems
Client_Process	None
IpServiceEndpoint	None
Client_ ConfigurationItem	None
Server_ ConfigurationItem	None

Topology Map

None

Supported Policy

None

Adapter Information

This job uses the **Running Software Dependencies via TCP Connection** adapter.

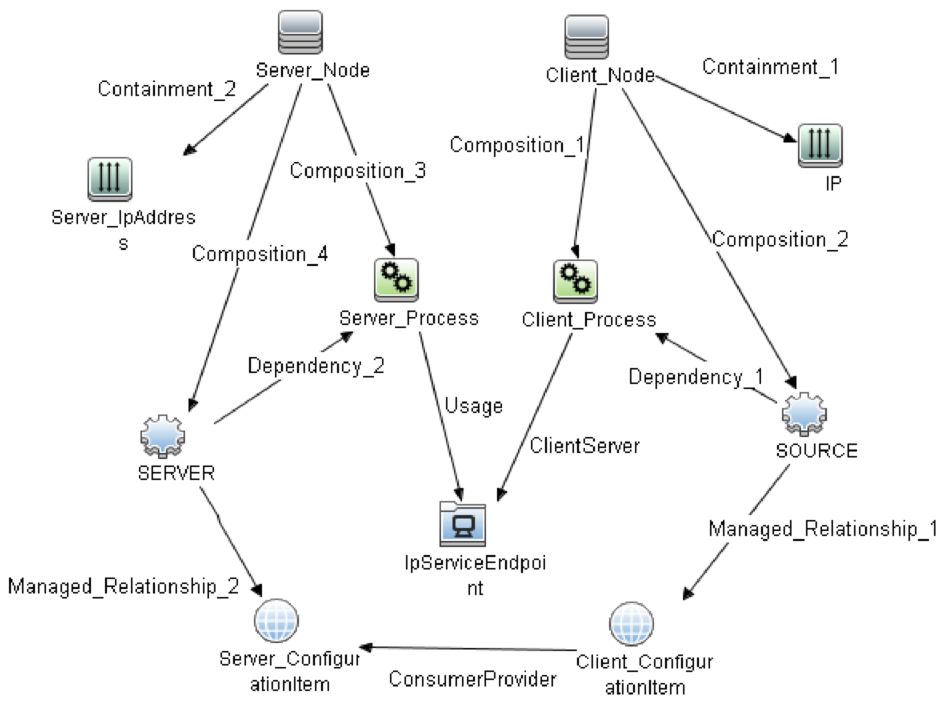
Adapter Type

Jython

Input CIT

RunningSoftware

Input TQL



Triggered CI Data

Name	Value
CLIENT_ID	${SOURCE.root_id:}
SERVER_ID	${SERVER.root_id:}

Workflow Steps

None

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

Chapter 5: Web Server

This chapter includes:

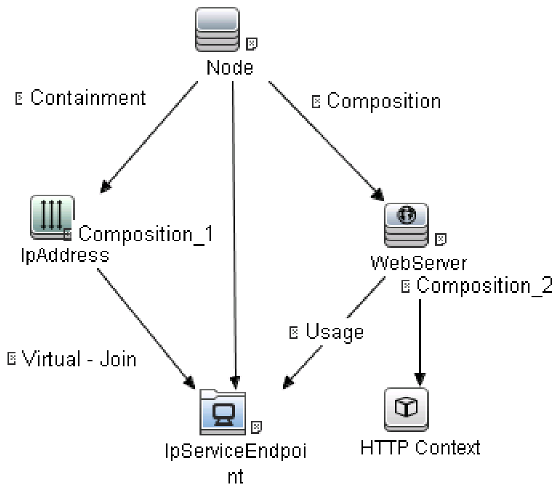
Web Server Dependencies via URL Job

This section includes details about the job.

Introduction

This job discovers and creates consumer-provider linkages between WebServer HTTP Context and BusinessElement via URL.

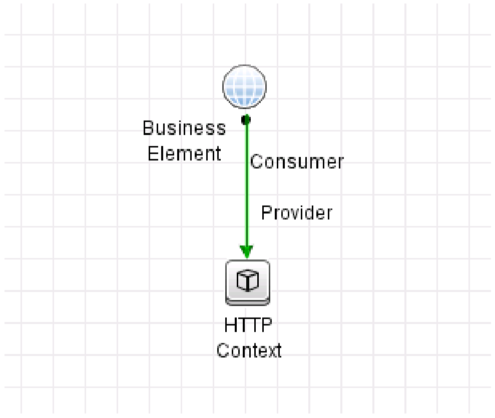
Trigger TQL



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	None
WebServer	None
HTTP Context	None

Topology Map

The **Web Server Dependencies via URL** topology is shown below.



Supported Policy

This job supports:

- Apache ProxyPass and ProxyPassReverse
- WebSphere plug-in for Apache, IIS, and IBM HTTP Server

Adapter Information

This job uses the **Web Server Dependencies via URL** adapter.

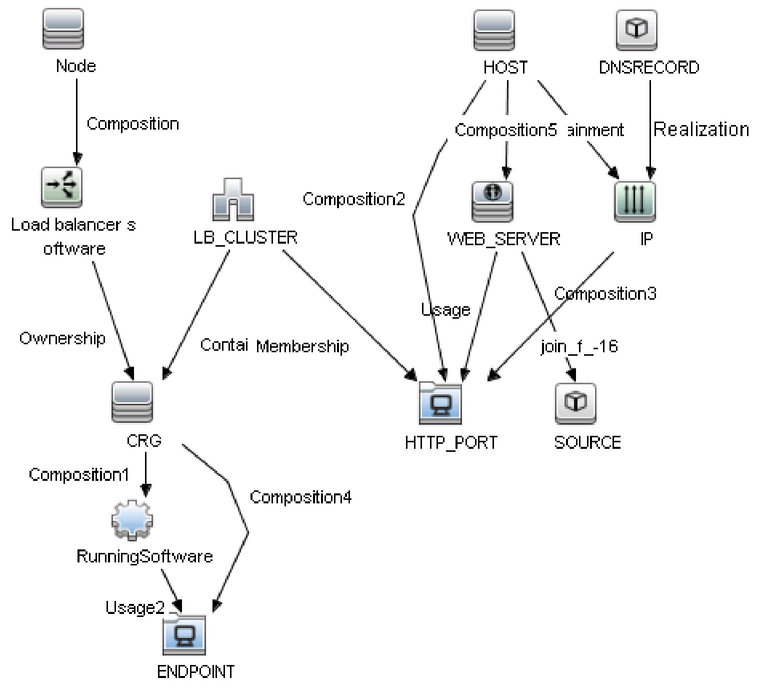
Adapter Type

Work Flow adapter

Input CIT

HTTP Context

Input TQL



Triggered CI Data

Name	Value
ALTER_DNSNAME	${IP.authoritative_dns_name:}
CONTEXT	${SOURCE.httpcontext_webapplicationcontext}
CLUSTER_RESOURCE_GROUP_ID	${CRG.root_id:}
DNS_NAME	${DNSRECORD.name:}
IP_ADDRESS	${IP.name}
PORT	${PORT.network_port_number}
PROVIDERTYPE	${SOURCE.root_class}
PROTOCOL	${SOURCE.applicationresource_type}

Name	Value
VIRTUAL_IP_ADDRESS	${ENDPOINT.bound_to_ip_address:NA}
VIRTUAL_PORT	${ENDPOINT.network_port_number:NA}

Workflow Steps

Step Name	Module	Failure-policy
Accurate Dependency Search	DependenciesDiscovery.py	Mandatory

Final Step Module
AccurateDependencyMapping.py

Discovered CITs

ConsumerProvider

Global Configuration Files

globalSettings.xml

Part 2: Cloud and Virtualization > Cloud

Chapter 6: Amazon Web Services Discovery

This chapter includes:

Overview

Amazon Web Services (AWS) is a collection of remote computing services (also called web services) that together make up a cloud computing platform, offered over the Internet by Amazon.com.

Amazon Web Services' offerings are accessed over HTTP, using Representational State Transfer (REST) and SOAP protocols.

AWS discovery shows the state and configuration of your cloud based on Amazon techonologies. The discovery of these low-level infrastructure services are supported:

- **EC2 (Compute)**

  Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity in the cloud. You define your virtual Amazon EC2 environment with the operating system, services, databases, and application platform stack required for your hosted application. Amazon EC2 provides a full management console and APIs to manage your compute resources.

- **RDS (Relational database)**

  Amazon Relational Database Service (Amazon RDS) is a web service that provides capacity for MySQL or Oracle deployments in the cloud, while managing time consuming tasks like backup, scaling, and patching.

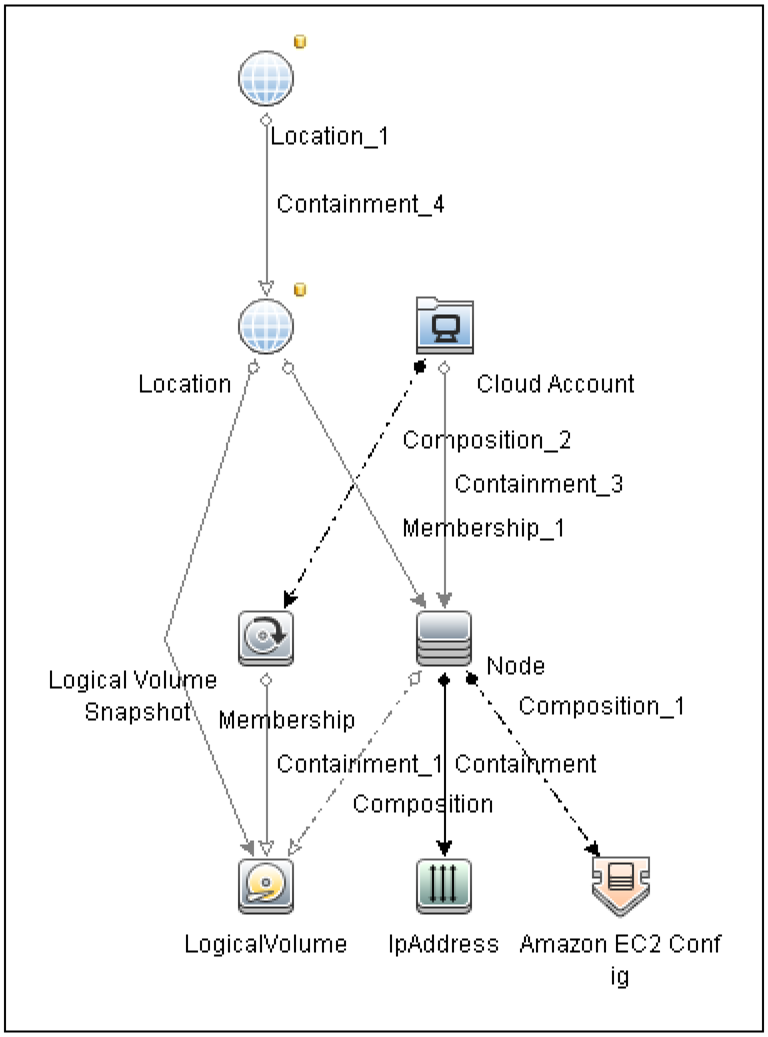
For communication with AWS, discovery uses Amazon SDK and IAM service for the authentication.

Topology

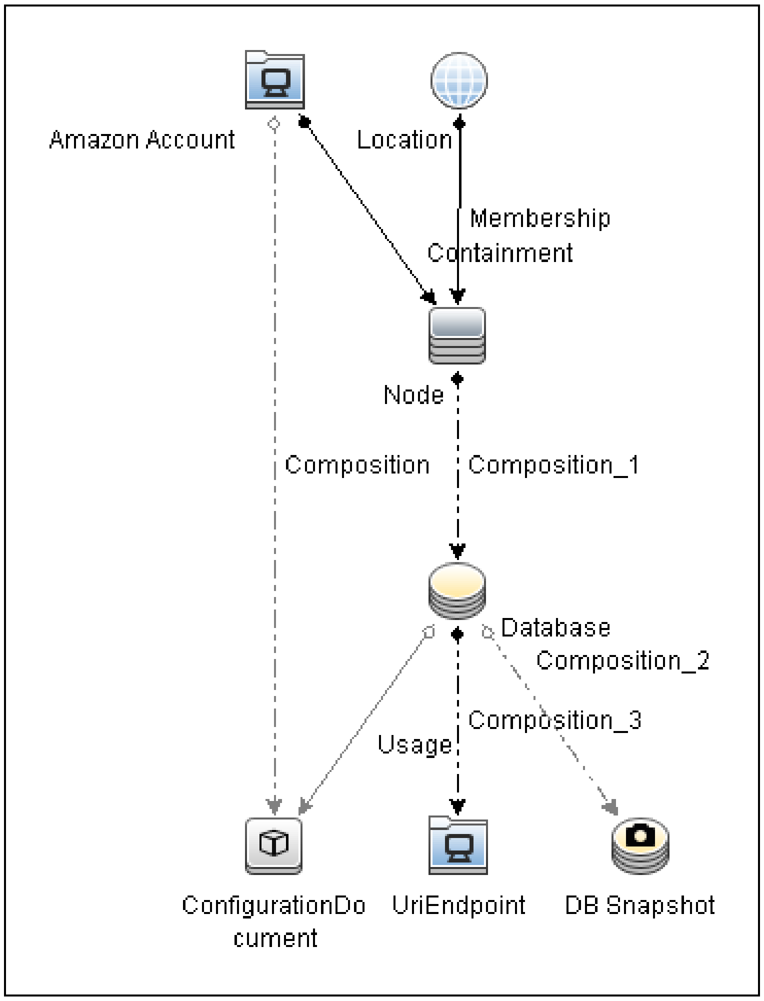
The following images display the topology of AWS discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 53.

Amazon EC2



Amazon RDS


Amazon Account
Location
Membership
Containment
Node
Composition
Composition_1
Database
Composition_2
Composition_3
Usage
ConfigurationDocument
UriEndpoint
DB Snapshot

How to Discover EC2 and RDS Services

This task describes how to discover two low-level AWS services, using a discovery protocol called **AWS Protocol**. This discovery process enables you to discover information about running node instances and their configuration (including information about AMI), corresponding block storage, and snapshots with information about regions and zones. All reported topology is in the scope of the Amazon account in which the discover user is registered.

This task contains the following steps:

- "Prerequisites - Probe IP address" below
- "Prerequisites - Credentials" below
- "Prerequisites - Driver setup" below
- "Run the discovery" on page 52

1. Prerequisites - Probe IP address

Discovery requires a probe with at least one IP address in range to trigger.

2. Prerequisites - Credentials

AWS discovery uses one of three types of access credentials used to authenticate requests to AWS services: **access keys**.

To represent AWS credentials in uCMDB you have to define: **AWS Protocol**.

Credential Value	AWS Protocol Name
Access Key ID	User Name
Secret Access Key	User Password

More information about **access keys** can be found here.

3. Prerequisites - Driver setup

Note: This step is required for each probe where you want to run AWS discovery.

a. Download the Amazon SDK for java from **http://aws.amazon.com/sdkforjava/**.

   The required version is 1.2.6 (referenced as ${VERSION} lately) or newer.

b. Unpack the zip file to a temporary folder; for example, ${AWS_TEMP_DIR}.

c. Create a folder **${PROBE_ROOT_DIR}/content/lib/aws/**, referred to as ${AWS_PROBE_DIR}.

d. Copy the third party library jars and SDK to ${AWS_PROBE_DIR}:

   ${AWS_TEMP_DIR}/lib/aws-java-sdk-${VERSION}.jar

   ${AWS_TEMP_DIR}/third-party/jackson-/.jar

   ${AWS_TEMP_DIR}/third-party/stax-ri-/.jar

   ${AWS_TEMP_DIR}/third-party/java-mail-/.jar

e. Configure the Data Flow Probe to load the driver jar files from ${PROBE_ROOT_DIR}/content/lib/aws/

f. Open **${PROBE_ROOT_DIR}/bin/WrapperEnv.conf**

g. In the **Environment global vars** section, add:

   ```
   set.aws=%CONTENT_LIB%/aws
   ```

   After the change, this **part** of the section should look like this:

   ```
   ...
   set.nnm=%CONTENT_LIB%/nnm
   set.aws=%CONTENT_LIB%/aws
   set.sap=%CONTENT_LIB%/sap
   ...
   ```

h. In the **Environment Discovery Path** section, first add:

   ```
   set.AWS_CLASSES=%aws%aws-java-sdk-${VERSION}.jar;%aws%httpcomponents-client-
   ${VERSION}*.jar;%aws%jackson-${VERSION}.jar;%aws%stax-
   ${VERSION}.jar;%aws%java-mail-${VERSION}.jar
   ```

   Note that you should replace **${VERSION}** with the exact jar version.

   Example:

```
set.AWS_CLASSES=%aws%aws-java-sdk-1.2.15.jar;%aws%httpcomponents-client-
4.1.1.jar;%aws%jackson-1.4.3.jar;%aws%stax-1.2.0.jar;%aws%java-mail-
1.4.3.jar
```

i. Also in the **Environment Discovery Path** section, add:

   ```
   ;%AWS_CLASSES%
   ```

   to the end of the line:

   ```
   set.COMMON_CLASSPATH
   ```

   After the change, the section should look like this:

   ```
   set.COMMON_CLASSPATH=%conf%;%XML_CLASSES%;%JYTHON_CLASSES%;%NNM_
   CLASSES%;%content_dll%;%FLOW_CLASSES%;%SAP_CLASSES%;%VMWARE_
   CLASSES%;%SYSTINET_CLASSES%;%CM_REDIRECT_CLASSES% ;%AWS_CLASSES%
   ```

j. Restart the Data Flow Probe.

4. **Run the discovery**

   Run the **AWS by Web Services** job.

AWS_by_WebServices Adapter

This section includes:

- "Input CIT" below
- "Triggered CI Data" on the next page
- "Used Scripts" on the next page
- "Discovered CITs" on the next page

Input CIT

Discovery Probe Gateway

Triggered CI Data

Name	Value
probeName	${SOURCE.name}

Used Scripts

- AWS_by_WebServices.py
- aws.py
- aws_rds.py
- aws_store.py
- entity.py
- db_platform.py
- db_builder.py
- db.py
- ec2.py
- iteratortools.py

Discovered CITs

- Amazon Account
- Amazon EC2 Config
- Composition
- ConfigurationDocument

- Containment
- Database
- DB Snapshot
- IpAddress
- Location
- LogicalVolume
- Logical Volume Snapshot
- Membership
- Node
- UriEndpoint
- Usage

Note: To view the topology, see "Topology" on page 48.

AWS by Web Services Job

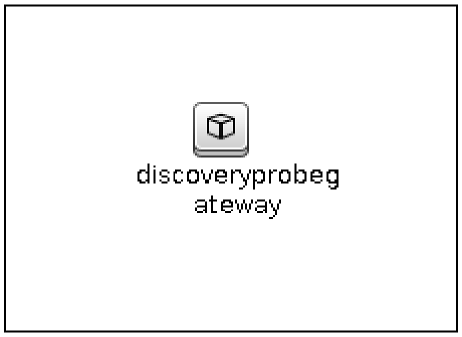
This section includes:

- "Adapter" below
- "Trigger Query" on the next page
- "Discovery Flow" on the next page

Adapter

This job uses the **AWS_by_WebServices** adapter.

Trigger Query



Discovery Flow

Discovering AWS, there is no IP address to trigger on, so the job starts against a probe where there is at least one IP address in the range. (This is a UCMDB work flow requirement.)

Before exploring any service, UCMDB needs to take information about the account the discovery user belongs to. This is done using IAM service; the user has an ARN (Amazon Resource Name) where the account ID is stored.

EC2 Service Discovery

- Get Regions and availability zones
- Get **running** instances; without this information all EBS discovery fails
- Get detailed information about EBS which is used as mapped devices for each running instance
- Get EBS Snapshot information for mapped EBS only
- Get AMI for each running instance; if AMI is not found, the corresponding instances are not reported to UCMDB
- Get Elastic IP information for each instance
- Data is immediately reported to UCMDB after discovery of each service

RDS Service Discovery

- Get database instances; without this information all RDS discovery fails
- Get all available engines to enrich information for every database instance server
- Get security and parameter groups to enrich available information in database instances
- Get database snapshots
- Data is immediately reported to UCMDB after discovery of each service

Chapter 7: VMware vCloud Discovery

This chapter includes:

Overview

VMware vCloud Director creates policy based virtual data centers by grouping together IT resources from multiple clusters.

The vCloud discovery process allows you to discover vCloud topology, including Organizations, Catalogs, Virtual Datacenters, vApps including Virtual Machines, vApps Templates, and Media.

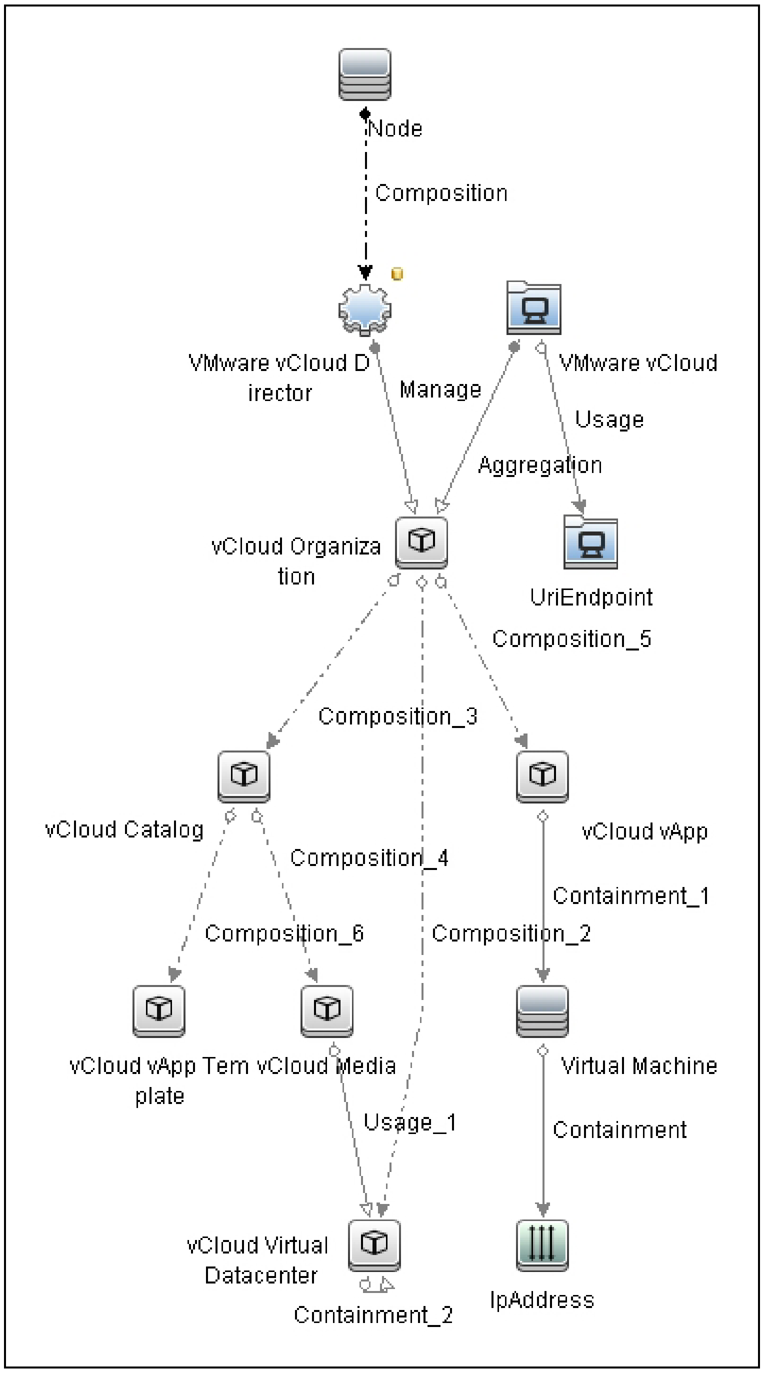
Supported Versions

VMware vCloud Discovery supports VMware vCloud Director Version 1.5 - 5.1.2.

Topology

The following image displays the topology of vCloud discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 63.



How to Discover vCloud by vCloud Director

This section describes how to discover the vCloud topology by discovering the vCloud Director application.

This task contains the following steps.

- "Prerequisites " below
- "Run the job" below

1. **Prerequisites**

   a. Shell connectivity to the host where the vCloud Director application runs.

   b. vCloud SDK jar files must be in the probe. See "How to Add vCloud SDK Dependencies to the Probe" on the next page.

   c. Define the following credentials:

      - **SSH** or **Telnet**
      - **vCloud**

        For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the job**

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **Host Connection by Shell** job to discover the target host and shell connectivity to it.

   c. Run the **Host Applications by Shell** job to discover applications of the target host, including the VMware vCloud Director application.

   d. Run the **vCloud Director by vCloud API** job to discover the vCloud topology.

How to Discover vCloud by URL

This section describes how to discover the vCloud topology using the URL of vCloud Director.

This task contains the following steps.

- "Prerequisites " below
- "Run the job" below

1. **Prerequisites**

    a. vCloud SDK jar files must be in the probe. See "How to Add vCloud SDK Dependencies to the Probe" below.

    b. Define the **vCloud** credential.

    For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the job**

    a. Run the **vCloud Director URL by vCloud API** job to discover the vCloud topology.

        i. Set the **baseUrl** parameter with the connection URL of the target vCloud Director.

        ii. After activating the job, manually add the probe which runs the discovery as an input CI.

How to Add vCloud SDK Dependencies to the Probe

To add vCloud SDK dependencies to the probe:

1. Download the VMware vCloud SDK archive from VMware community site:

    http://communities.vmware.com/community/vmtn/developer/forums/vcloudsdkjava

    The recommended version is 5.1.

2. Copy the following jar files to the **%PROBE_ROOT%\content\lib** folder:

- **SDK-<version>\rest-api-schemas-<version>.jar**
- **SDK-<version>\vcloud-java-sdk-<version>.jar**

3. Restart the probe.

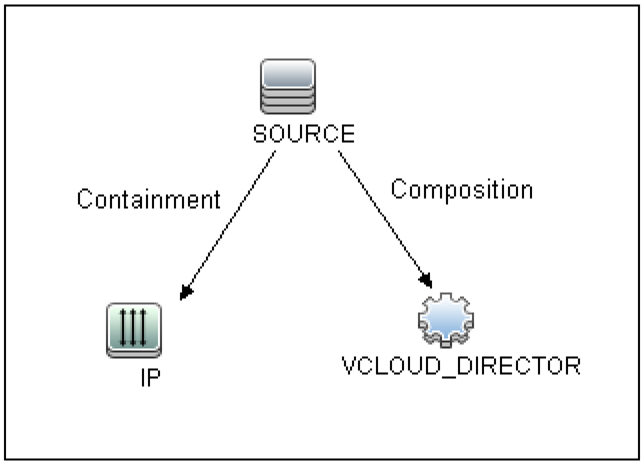
vCloud_Director_by_vCloud_API Adapter

This section contains details about the adapter.

Input CIT

Node

Input Query



Node Name	Condition
SOURCE	None
IP	NOT IP Probe Name Is null
VCLOUD_DIRECTOR	DiscoveredProductName Equal VMware vCloud Director

Triggered CI Data

Name	Value
ip_addresses	${IP.name}
vCloudDirectorId	S{VCLOUD_DIRECTOR.root_id}

Used Scripts

- vcloud.py
- vcloud_director_by_vcloud_api.py
- vcloud_discover.py
- vcloud_report.py

Discovered CITs

- Aggregation
- Composition
- Containment
- Interface
- IpAddress
- Manage
- Node
- UriEndpoint
- Usage
- vCloud Catalog

- vCloud Media
- vCloud Organization
- vCloud vApp
- vCloud vApp Template
- vCloud Virtual Datacenter
- VMware vCloud
- VMware vCloud Director

Note: To view the topology, see "Topology" on page 59.

Parameters

Name	Description
reportPoweredOffVms	When set to **True**, powered off virtual machines are reported. When set to **False**, powered off virtual machines are not reported. **Default**: False

vCloud_Director_URL_by_vCloud_API Adapter

This section contains details about the adapter.

Input CIT

Discovery Probe Gateway

Used Scripts

- vcloud.py

- vcloud_director_url_by_vcloud_api.py
- vcloud_discover.py
- vcloud_report.py

Discovered CITs

- Aggregation
- Composition
- Containment
- Interface
- IpAddress
- Manage
- Node
- UriEndpoint
- Usage
- vCloud Catalog
- vCloud Media
- vCloud Organization
- vCloud vApp
- vCloud vApp Template
- vCloud Virtual Datacenter
- VMware vCloud
- VMware vCloud Director

Note: To view the topology, see "Topology" on page 59.

Parameters

Name	Description
baseUrl	The connection URL of the target vCloud Director
reportPoweredOffVms	When set to **True**, powered off virtual machines are reported. When set to **False**, powered off virtual machines are not reported. **Default**: False

vCloud Director by vCloud API Job

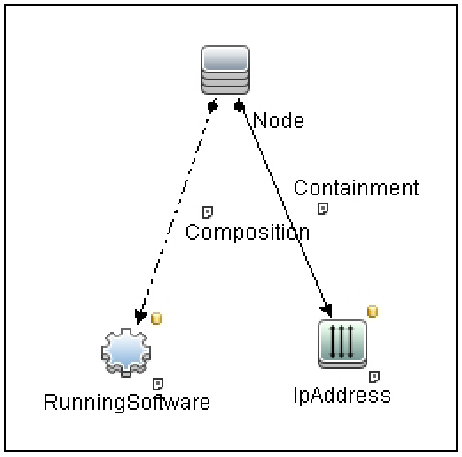
This section contains details about the job.

Adapter

This job uses the vCloud_Director_by_vCloud_API adapter.

Trigger Query

vcloud_director_on_host_with_ip



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
RunningSoftware	DiscoveredProductName Equal VMware vCloud Director

Parameters

Parameters are not overridden by default and use the values from the adapter.

vCloud Director URL by vCloud API Job

This section contains details about the job.

- **Adapter**

  This job uses the vCloud_Director_URL_by_vCloud_API adapter.

- **Trigger Query**

  None

- **Parameters**

  Parameters are not overridden by default and use the values from the adapter.

Troubleshooting and Limitations

A virtual machine which is part of vApps, and has neither a MAC address available nor a connected network adapter, is not reported.

Part 3: Cloud and Virtualization > Virtualization

Chapter 8: HP IVM Discovery

This chapter includes:

Overview

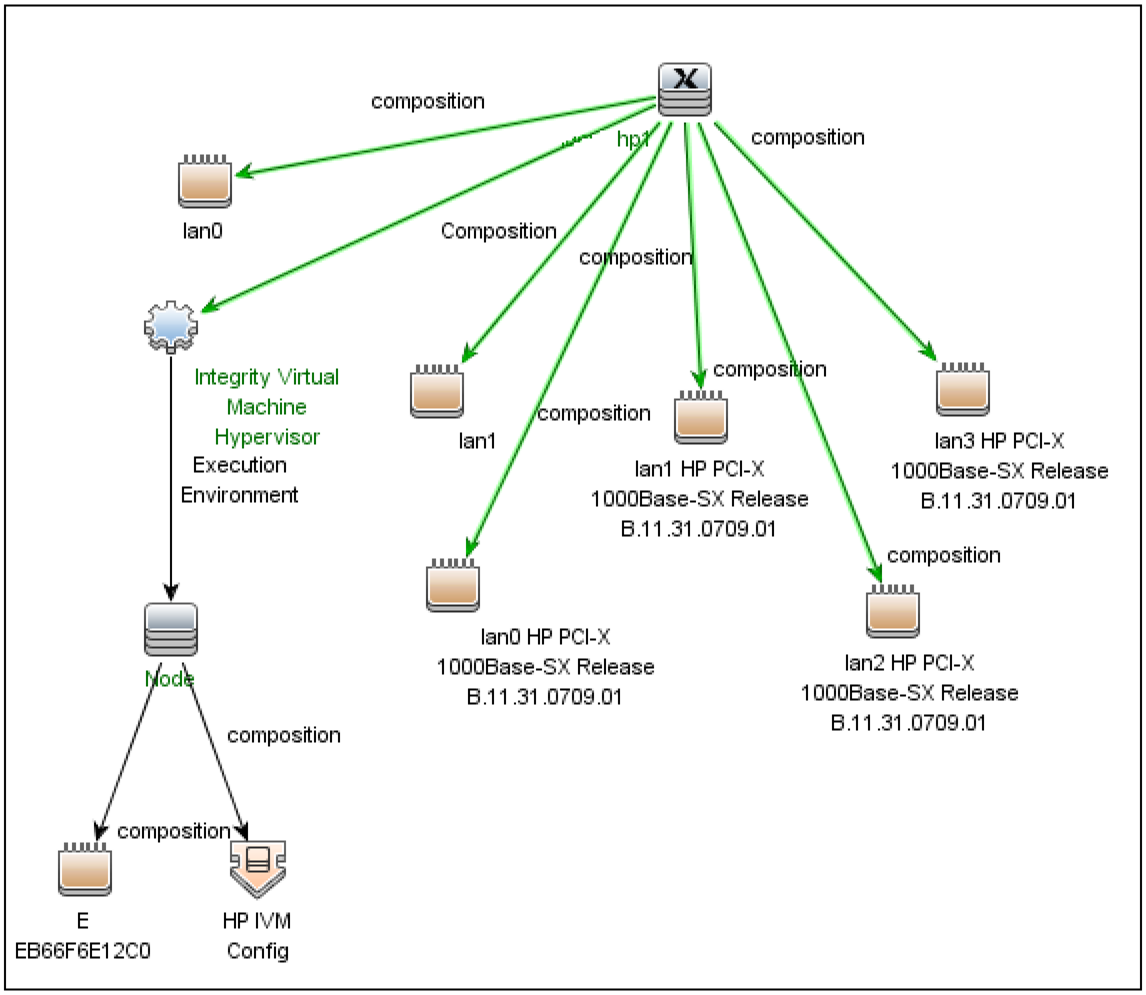
Integrity Virtual Machines (IVM) is HP software that allows multiple virtual machines (such as the HP Integrity line) to run concurrently on any Itanium server running HP-UX. HP IVM is part of HP's Virtual Server Environment suite.

Supported Versions

HP IVM Discovery supports HP IVM version B.06.10.05.

Topology

The HP IVM topology is shown below.


hp1
composition
lan0
Composition
composition
Integrity Virtual Machine Hypervisor
Execution Environment
lan1
composition
composition
lan1 HP PCI-X 1000Base-SX Release B.11.31.0709.01
lan3 HP PCI-X 1000Base-SX Release B.11.31.0709.01
lan0 HP PCI-X 1000Base-SX Release B.11.31.0709.01
lan2 HP PCI-X 1000Base-SX Release B.11.31.0709.01
Node
composition
E EB66F6E12C0
HP IVM Config

How to Discover HP IVM Topology

This section describes how to discover the topology managed by HP IVM.

Prerequisites

- Ensure that there is shell connectivity with a running IVM machine.
- Set up SSH protocol credentials. For more information on this, see the section explaining SSH protocol credentials in *HP UCMDB Universal Discovery Content Guide - Supported Content*.

Run HP IVM Topology Discovery

Run HP IVM Topology Discovery by executing the following jobs:

1. **Range IPs by ICMP** (discover the target IPs)
2. **Host Connection by Shell** (discovers the target host, and shell connectivity to it)
3. **HP IVM by Shell** (discovers the virtualization environment managed by HP IVM)

HP IVM by Shell Job

This job discovers the HP IVM topology.

Adapter

ID: HP Integrity Virtual Machine by Shell

Display Name: HP Integrity Virtual Machine by Shell

Trigger TQL

CIT Name	Condition
Computer	ExtendedOsFamily Equal hp_ux
IpAddress	Not Ip Probe Name Is Null
Shell	NOT Reference to credentials dictionary entry is Null

Discovery Flow

The discovery flow for the HP IVM by Shell Job is as follows:

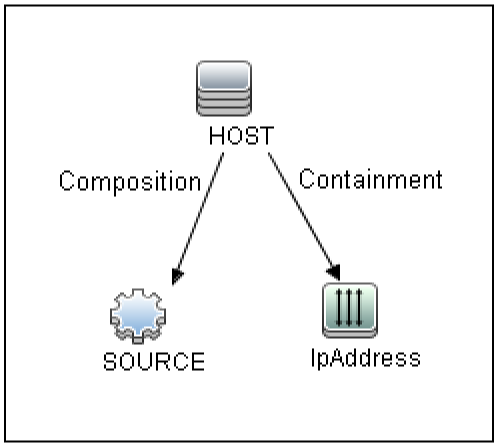
1. Get the current status of the running virtual machines using the command **hpvmstatus -V**.
2. Get VM configuration data using the command **hpvmstatus -d -P <vm_name>**.
3. Get version related information using the command **hpvminfo -v**.

HP Integrity Virtual Machine by Shell Adapter

Input CIT

Shell

Input TQL



CIT Name	Condition
HOST	NOT CI Type Equal nt

Triggered CI Data

Name	Value
Protocol	SOURCE.root_class

Name	Value
credentialsId	SOURCE.credentials_id
hostId	HOST.root_id
ip_address	SOURCE.application_ip

Used Scripts

- ivm.py
- ivm_by_shell.py
- ivm_discoverer.py

Discovered CITs

- Composition
- ExecutionEnvironment
- HP IVM Config
- Interface
- Node
- Virtualization Layer Software

Note: To view the topology, see "Topology" on page 71.

Parameters

```
reportHostNameAsVmName = false
```

Chapter 9: HP Partitioning Solution Discovery

This chapter includes:

Overview

- **HP nPartitions**

  Cell-based HP servers enable you to configure a single server complex as one large system or as multiple smaller systems by configuring **nPartitions**. Each nPartition defines a subset of server hardware resources to be used as an independent system environment. An nPartition includes one or more cells assigned to it (with processors and memory) and all I/O chassis connected to those cells. All processors, memory, and I/O in an nPartition are used exclusively by software running in the nPartition. Thus, each nPartition has its own system boot interface, and each nPartition boots and reboots independently. Each nPartition provides both hardware and software isolation, so that hardware or software faults in one nPartition do not affect other nPartitions within the same server complex. You can reconfigure nPartition definitions for a server without physically modifying the server hardware configuration by using the HP software-based nPartition management tools.

- **HP vPartitions**

  vPars is a Virtual Partitions product that enables you to run multiple instances of HP-UX simultaneously on one hard partition by dividing that hard partition further into virtual partitions. Each virtual partition is assigned its own subset of hardware, runs a separate instance of HP-UX, and hosts its own set of applications. Because each instance of HP-UX is isolated from all other instances, vPars provides application and Operating System (OS) fault isolation. Each instance of HP-UX can have different patches and a different kernel.

Supported Versions

This discovery supports vPars A.03.xx, A.04.xx, and A.05.xx versions.

This package has been verified on cellular systems with vPars running an HP-UX operating system.

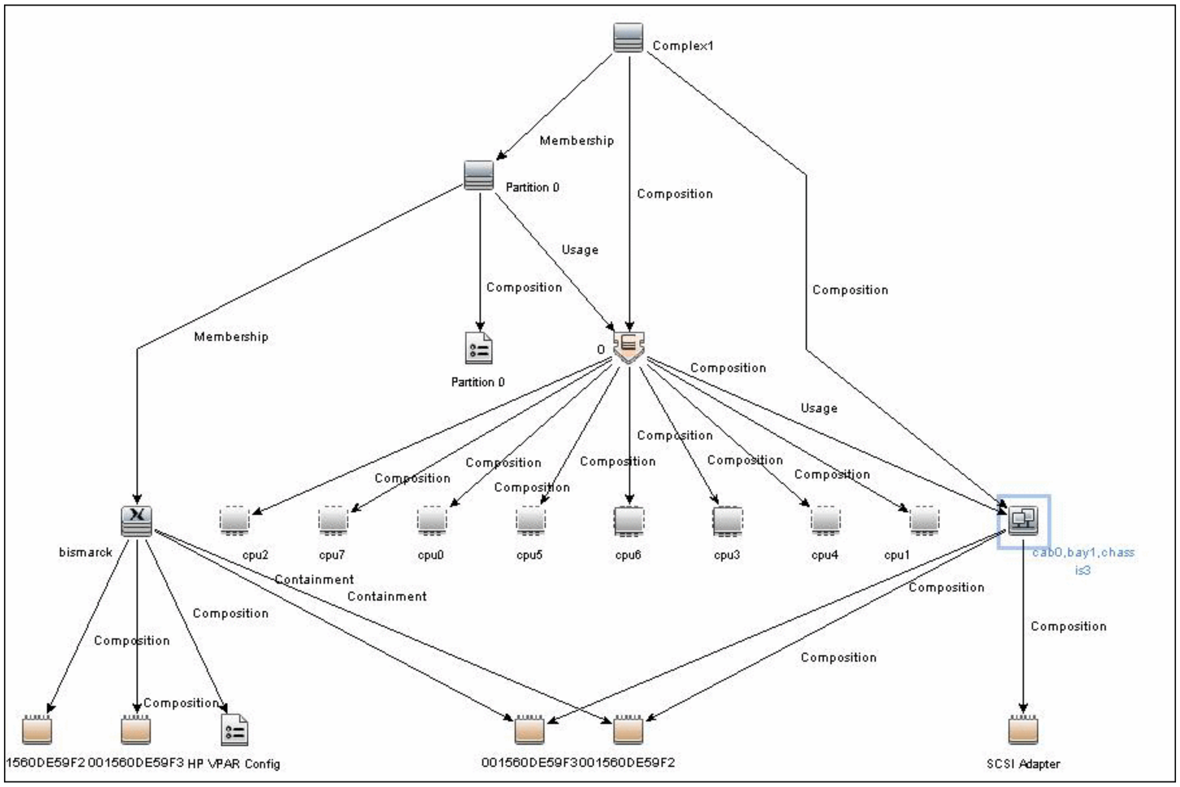
This discovery supports HP Blade based complexes.

Topology

This section includes:

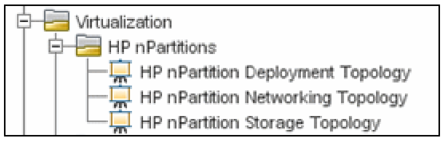
- "HP vPars and nPars Topology" below
- "HP nPartitions Topology Views" below

HP vPars and nPars Topology

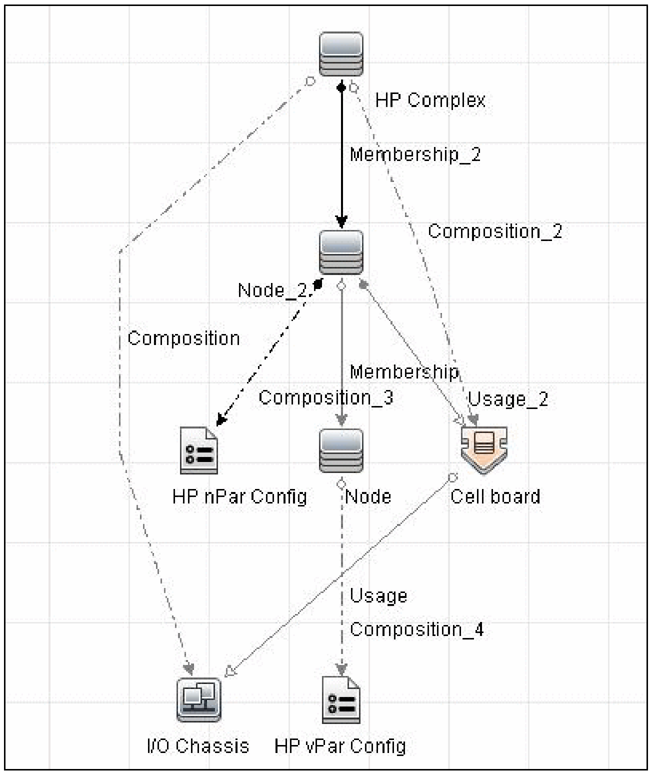


HP nPartitions Topology Views

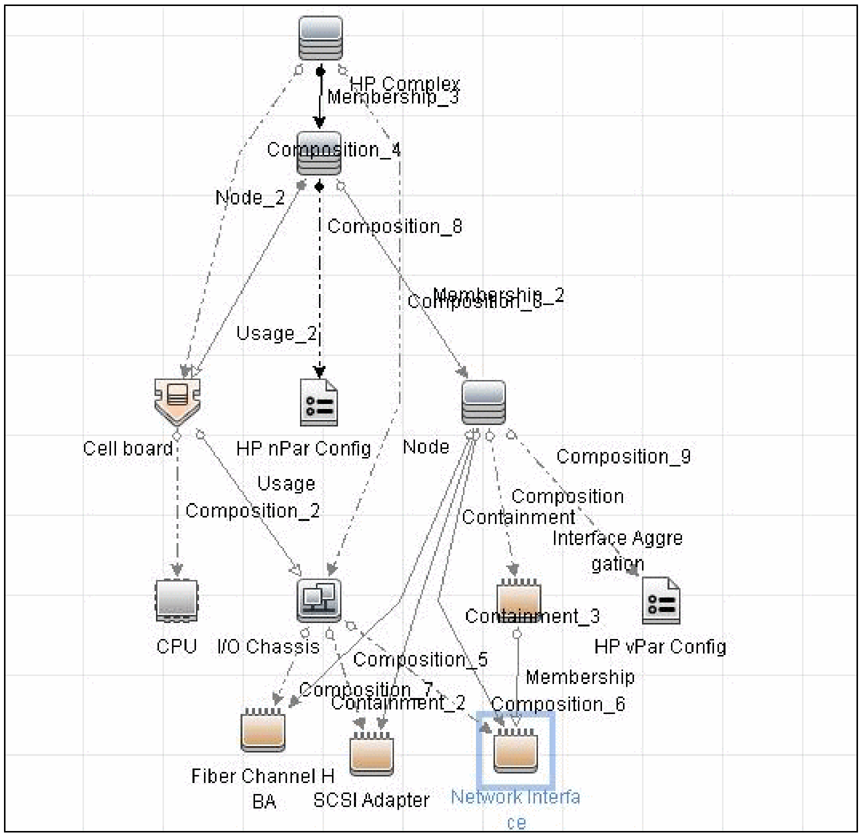
HP nPartitions topology is represented by the following views under the Virtualization module:


Virtualization
HP nPartitions
HP nPartition Deployment Topology
HP nPartition Networking Topology
HP nPartition Storage Topology

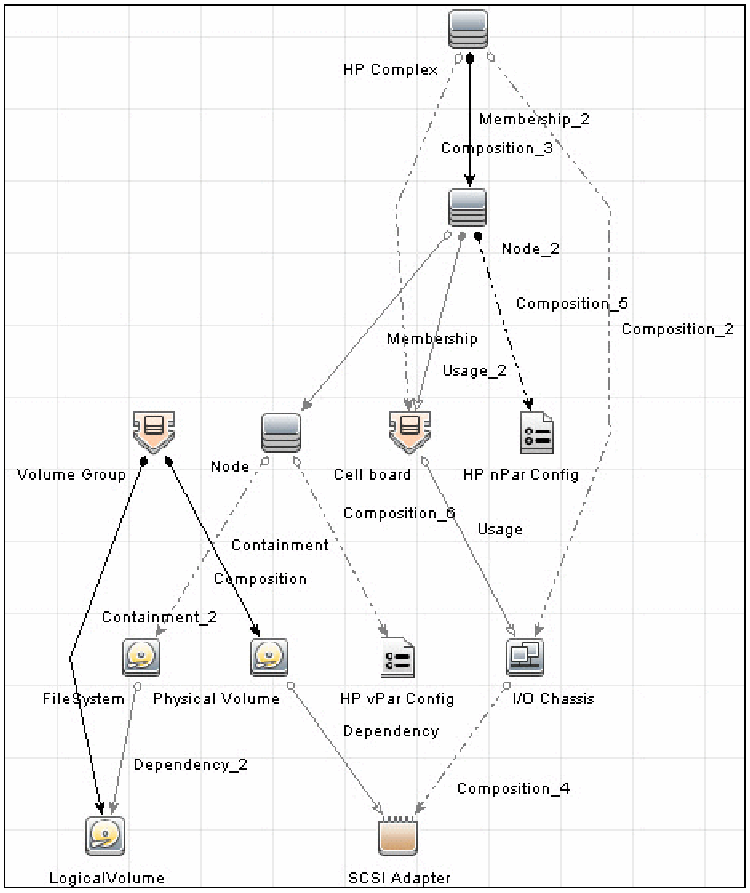
- **HP nPartition Deployment Topology View**
  This view represents the basic virtualization deployment, containing nPars, vPars, cells, and I/O chassis only.



- **HP nPartition Networking Topology View**
  This view represents the Networking aspect of the nPartition deployment including the relations between I/O devices of vPars and their physical locations on the I/O chassis.



- **HP nPartition Storage Topology View**
  This view reflects the storage aspect of the HP nPartitions system including the relations between file systems and logical volumes.



How to Discover HP vPars and nPars

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Confirm that Shell credentials are set up on the Probe.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow*

Management Guide.

a. Run the **Range IPs by ICMP** job.

b. Run the **Host Connection by Shell** job.

c. Run the **HP nPars and vPars by Shell** job.

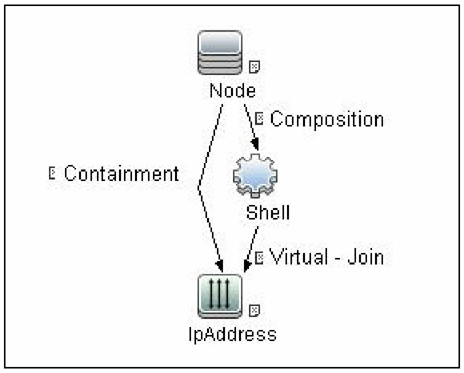
For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

HP nPars and vPars by Shell Job

This section includes details about the job.

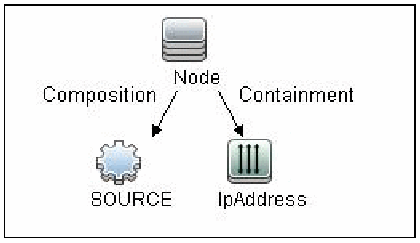
Trigger Query

Note: The host_shell name is also used by the Host Applications by Shell and Host Resources by Shell jobs.



Adapter

- The Input Query for the hp_npar_by_shell Adapter



Created/Changed Entities

New Classes

- hp_complex
- cell_board
- io_chassis
- hp_npar_config
- hp_vpar_config

End1	Relationship Type	End2
node	containment	fchba
node	containment	interface
node	containment	scsi_adapter
cell_board	composition	cpu
cell_board	composition	memory
hp_complex	composition	io_chassis
io_chassis	composition	fchba
io_chassis	composition	interface
io_chassis	composition	scsi_adapter
cell_board	usage	io_chassis

End1	Relationship Type	End2
node	usage	cell_board
node	usage	fchba
node	usage	interface

Discovered CITs

- Composition
- Containment
- Cpu
- Dependency
- Fibre Channel HBA
- FileSystem
- HP Complex
- HP nPar Config
- HP vPar Config
- I/O Chassis
- Interface
- Interface Aggregation
- LogicalVolume
- Membership
- Node
- Physical Volume

- SCSI Adapter
- Usage
- Volume Group

Discovery Mechanism

This section includes the following commands:

Verify Discovery on the vPartition

Goal	1. To verify if discovery has connected to the vPartition. 2. To verify that further commands produce supported output.
Command	vparstatus -V
Output	Version 2.0
Values taken	1. 2.0. The version of the **vparstatus** executable 2. Return code
Comment	Supported versions of output are 2.0 and 1.3

Verify Discovery on the nPartition

Goal	To understand if discovery has connected to the partitionable server.
Command	parstatus -s
Output	None
Values taken	Return code
Comment	If return code is **0**, discovery has connected to the partitionable system

Get Information about Complex

Goal	To retrieve properties of the **HP Complex** CIT.
Command	parstatus -X

Output rp8420	[Complex] Complex Name : Complex 01 Complex Capacity Compute Cabinet (4 cell capable) : 1 Active GSP Location : cabinet 0 Model : 9000/800/rp8420 Serial Number : DEH45419K0 Current Product Number : A6912A Original Product Number : A6912A Complex Profile Revision : 1.0 The total number of Partitions Present : 2
Output rx8640	[Complex] Complex Name : Complex 01 Complex Capacity Compute Cabinet (4 cell capable) : 1 Active MP Location : cabinet 0 Original Product Name : server rx8640 Original Serial Number : DEH4831H1Y Current Product Order Number : AB297A OEM Manufacturer : Complex Profile Revision : 1.0 The total number of partitions present : 1

Output sx3000	`[Complex]`
	
	`Complex Name: Complex 1`
	
	`Complex UUID: ab111111-2222-3333-4444-555555555555`
	
	`Complex Capacity Compute Enclosure: 1 IO Enclosure: 8`
	
	`Monarch OA Location: 1`
	
	`Model: N/A`
	
	`Original Serial Number: AB11111111`
	
	`Current Product Order Number: CD2222`
	
	`OEM Manufacturer: N/A`
	
	`Total number of partitions present: 2`

Values taken	• Complex Name > name • Serial number/Original Serial Number > serialnumber, hostkey
Comment	HP Complex CIT derives from the Host CIT

List General Information About All Cells

Goal	To retrieve the list of names of all Cells of all Cabinets in the Complex.
Command	parstatus -C -M

Output rp8420	`cell:cab0,cell0:active core :8/0/8 :48.0/ 0.0:cab0,bay0,chassis0 :yes :yes :0`
	
	`cell:cab0,cell1:active core :4/0/8 :32.0/ 0.0:cab0,bay0,chassis1 :yes :yes :1`
	
	`cell:cab0,cell2:active base :8/0/8 :40.0/ 0.0:- :no :yes :0`
	
	`cell:cab0,cell3:active base :4/0/8 :32.0/ 0.0:- :no :yes :1`

Output rx8640	`cell:cab0,cell0:Active Core :8/0/8 :80.0/0.0 :cab0,bay0,chassis0 :yes :yes :0`
	
	`cell:cab0,cell1:Active Base :8/0/8 :80.0/0.0 :cab0,bay0,chassis1 :yes :yes :0`
	
	`cell:cab0,cell2:Active Base :4/0/8 :64.0/0.0 :- :no :yes :0`
	
	`cell:cab0,cell3:Absent :- :- :- :- :-`

Values taken	The names of the cells
Comment	The cell names are then used to retrieve detailed information about each cell.

List Detailed Information About Each Cell

Goal	To retrieve the properties of the Cell CIs and corresponding CPU and Memory CIs.
Command	parstatus -v -c <cell_number>

Output rp8420	`[Cell]`
	`Hardware Location : cab0,cell0`
	`Global Cell Number : 0`
	`Actual Usage : active core`
	`Normal Usage : base`
	`Connected To : cab0,bay0,chassis0`
	`Core Cell Capable : yes`
	`Firmware Revision : 24.1`
	`Failure Usage : activate`
	`Use On Next Boot : yes`
	`Partition Number : 0`
	`Partition Name : db01_ap02_db03_db04`
	`[CPU Details]`
	`Type : 88E0`
	`Speed : 1100 MHz`
	`CPU Status`
	`=== ======`
	` 0 ok`
	` 1 ok`
	` 2 ok`
	` 3 ok`
	` 4 ok`
	` 5 ok`
	` 6 ok`
	` 7 ok`
	`CPUs`
	`===========`
	`OK : 8`
	`Deconf : 0`
	`Max : 8`
	`[Memory Details]`

	`DIMM Size (MB) Status`
	
	`==== ========= =========`
	
	`0A 4096 ok`
	
	`4A 4096 ok`
	
	`0B 4096 ok`
	
	`4B 4096 ok`
	
	`1A 4096 ok`
	
	`5A 4096 ok`
	
	`1B 4096 ok`
	
	`5B 4096 ok`
	
	`2A 4096 ok`
	
	`2B 4096 ok`
	
	`3A 4096 ok`
	
	`3B 4096 ok`
	
	`Memory`
	
	`=========================`
	
	`DIMM OK : 12`
	
	`DIMM Deconf : 0`
	
	`Max DIMMs : 16`
	
	`Memory OK : 48.00 GB`
	
	`Memory Deconf : 0.00 GB`

Output rx8640	[Cell] Hardware Location : cab0,cell0 Global Cell Number : 0 Actual Usage : Active Core Normal Usage : Base Connected To : cab0,bay0,chassis0 Core Cell Capable : yes Firmware Revision : 9.48 Failure Usage : Normal Use On Next Boot : yes Partition Number : 0 Partition Name : db10_ap13_ap14_db15_db16_ap17_ap18_ap20 Requested CLM value : 0.0 GB Allocated CLM value : 0.0 GB Cell Architecture Type : Itanium(R)-based CPU Compatibility : CDH-640 Hyperthreading Capable : yes [CPU Details] Type : FFFF Speed : 1598 MHz CPU Status === ====== 0 OK 1 OK 2 OK 3 OK 4 OK 5 OK 6 OK 7 OK CPUs

	`===========`
	`OK : 8`
	`Deconf : 0`
	`Max : 8`
	`[Memory Details]`
	`DIMM Size (MB) Status`
	`==== ========= =========`
	`3A 8192 OK`
	`3B 8192 OK`
	`1A 8192 OK`
	`1B 8192 OK`
	`4A 8192 OK`
	`4B 8192 OK`
	`0A 8192 OK`
	`0B 8192 OK`
	`2A 8192 OK`
	`2B 8192 OK`
	`Memory`
	`=========================`
	`DIMM OK : 10`
	`DIMM Deconf : 0`
	`Max DIMMs : 16`
	`Memory OK : 80.00 GB`
	`Memory Deconf : 0.00 GB`

Values taken	Global Cell Number > name	
	Hardware Location > hardware_path	
	Actual Usage > is_core	If value of **Actual Usage** contains the word **Core**
	Core Cell Capable > core_capable	Convert **yes/no** to Boolean
	Requested CLM value > requested_clm_value	• This parameter does not exist for rp8420 servers • Need to convert GB to MB
	Allocated CLM value > allocated_clm_memory	• This parameter does not exist for rp8420 servers • Need to convert GB to MB
	Use On Next Boot > use_on_next_boot	Convert **yes/no** to Boolean
	Failure Usage > failure_usage	
	Firmware Revision > firmware_revision	
	Cell Architecture Type > architecture_type	This value does not exist for rp8420 servers
	CPU Compatibility > cpu_compatibility	This value does not exist for rp8420 servers
	Hyperthreading Capable > is_hyperthreading_capable	Convert **yes/no** to Boolean

Values taken *(cont'd)*	`CPUs`
	`===========`
	`OK : 8`
	`Deconf : 0`
	`Max : 8`
		deconf_cpu_number: 0
		max_cpu_number: 8	

	`Memory`
	`=============`
	`DIMM OK : 10`
	`DIMM Deconf : 0`
	`Max DIMMs : 16`
	`Memory OK : 80.00 GB`
	`Memory Deconf : 0.00 GB`
		memory_amount: 80.00 GB
		deconf_memory: 0.00 GB
		max_dimms:16
		deconfigured_dimms: 0
			Need to convert GB to MB

Comment	The Memory CI is not created for UCMDB 9.x since there is no such CIT. The partition number is used to connect the cell to the nPartition (represented as a host).		

Get Information About I/O Chassis

Goal	To retrieve the data of all I/O chassis in the Complex (including I/O extension cabinets).	
Command	parstatus -I -M	

Output rp8420	`chassis:cab0,bay0,chassis0 :active :yes :cab0,cell0:0`
	`chassis:cab0,bay0,chassis1 :active :yes :cab0,cell1:1`	

Output rx8640	`chassis:cab0,bay0,chassis0 :Active :yes :cab0,cell0:0`
	`chassis:cab0,bay0,chassis1 :Active :yes :cab0,cell1:0`	

Values taken	name: cab0,bay0,chassis0	
	usage: Active	
	is_core: yes	To convert to Boolean values.
Comment	The Cell hardware path is used to connect the chassis to the Cell.	

Get the List of Names of the nPartitions on the System

Goal	To retrieve the list of the nPartition numbers configured on the system.
Command	parstatus -P -M

Output rp8420	`partition: 0 :active : 2 : 1 :cab0,cell0:db01_ap02_db03_db04`
	`partition: 1 :active : 2 : 1 :cab0,cell1:wdb1_wdb4`

Output rx8640	`partition:0 :Active :3 :2 :cab0,cell0:db10_ap13_ap14_db15_db16_ap17_`
Values taken	The list of nPartition numbers
Comment	These numbers are used to retrieve detailed information about each nPartition.

Get Detailed Information About nPartition

Goal	To retrieve detailed information for each nPartition and create a Host, connected to the Cells and to the **HP nPar Config** CI.
Command	parstatus -v -p <npartition_number>

Output rp8420	[Partition] Partition Number : 0 Partition Name : db01_ap02_db03_db04 Status : active IP address : 0.0.0.0 Primary Boot Path : 0/0/0/2/0.6.0 Alternate Boot Path : 0/0/0/2/1.2.0 HA Alternate Boot Path : 0/0/0/3/0.6.0 PDC Revision : 24.1 IODCH Version : 88E0 CPU Speed : 1100 MHz Core Cell : cab0,cell0 [Cell] CPU Memory Use OK/ (GB) Core On Hardware Actual Deconf/ OK/ Cell Next Par Location Usage Max Deconf Connected To Capable Boot Num ========== ============ ======= === cab0,cell0 active core 8/0/8 48.0/ 0.0 cab0,bay0,chassis0 yes yes 0 cab0,cell2 active base 8/0/8 40.0/ 0.0 - no yes 0 [Chassis] Core Connected Par Hardware Location Usage IO To Num =================== ============ == === cab0,bay0,chassis0 active yes cab0,cell0 0

Output rx8640	`[Partition]`
	`Partition Number : 0`
	`Partition Name : db10_ap13_ap14_db15_db16_ap17_ap18_ap20`
	`Status : Active`
	`IP Address :`
	`Primary Boot Path : 0/0/8/1/0/4/0.8.0.255.0.12.0`
	`Alternate Boot Path : 0/0/8/1/0/4/1.8.0.255.0.13.0`
	`HA Alternate Boot Path :`
	`PDC Revision : 9.48`
	`IODCH Version : ffff`
	`Cell Architecture : Itanium(R)-based`
	`CPU Compatibility : CDH-640`
	`CPU Speed : 1598 MHz`
	`Core Cell : cab0,cell0`
	`Core Cell Choice [0] : cab0,cell0`
	`Total Good Memory Size : 224.0 GB`
	`Total Interleave Memory: 224.0 GB`
	`Total Requested CLM : 0.0 GB`
	`Total Allocated CLM : 0.0 GB`
	`Hyperthreading Enabled : no`
	`[Cell]`
	` CPU Memory Use`
	` OK/ (GB) Core On`
	`Hardware Actual Deconf/ OK/ Cell Next Par`
	`Location Usage Max Deconf Connected To Capable Boot Num`
	`========== ============ ======= ========`
	`cab0,cell0 Active Core 8/0/8 80.0/0.0 cab0,bay0,chassis0 yes yes 0`
	`cab0,cell1 Active Base 8/0/8 80.0/0.0 cab0,bay0,chassis1 yes yes 0`
	`cab0,cell2 Active Base 4/0/8 64.0/0.0 - no yes 0`

```
Notes: * = Cell has no interleaved memory.

[Chassis]

 Core Connected Par

Hardware Location Usage IO To Num

=================== ============ ====

cab0,bay0,chassis0 Active yes cab0,cell0 0

[Chassis]

 Core Connected Par

Hardware Location Usage IO To Num

=================== ============ ==== ========== ===

cab0,bay0,chassis1 Active yes cab0,cell1 0
```

Values taken	Host (nPartition)	
	hostkey	Host key is composed of nPartition name and Complex Serial number
	Partition Name > tname	
	HP nPar Config	
	Constant "nPar Config" > name	
	Partition Name > npar_name	
	Status > npar_status	
	PDC Revision > pdc_revision	
	Hyperthreading Enabled > hyperthreading_mode	This value does not exist on the rp8420 servers
	Partition Number > partition_number	
	Primary Boot Path > primary_boot_path	
	Alternate Boot Path > alternate_boot_ path	

Get the Name of the Current vPartition

Goal	To retrieve the name of the current vPartition.
Command	vparstatus -w -M
Output	`doidb01`
Values taken	The name of the vPartition that discovery has connected to.
Comment	The list includes detailed information for the current vPartition only. It is possible to retrieve detailed information about all vPartitions on the nPartition, but it is not possible to retrieve their IP addresses and/or lower MAC address to create a host in UCMDB.

Get Detailed Information About vPartition

Goal	To retrieve detailed information about vPartition and create **Host** and **HP vPar Config** CIs.
Command	vparstatus -v -p <vpartition_name>

Output rp8420	
	```
	[Virtual Partition Details]
	Name: doidb01
	State: Up
	Attributes: Dynamic,Autoboot,Nosearch
	Kernel Path: /stand/vmunix
	Boot Opts: -lq
	[CPU Details]
	Min/Max: 3/16
	Bound by User [Path]: 0.15
	 0.16
	 0.17
	Bound by Monitor [Path]:
	Unbound [Path]: 2.14
	 2.15
	[IO Details]
	 0.0.12
	 0.0.14
	 0.0.12.1.0.4.0.8.0.255.0.0.0
	 0.0.14.1.0.4.0.8.0.255.0.1.0
	 0.0.12.1.0.4.0.111.128.19.4.0.0
	 0.0.12.1.0.4.0.111.88.19.5.0.0 BOOT
	 0.0.14.1.0.4.0.112.88.19.5.0.0, ALTBOOT
	[Memory Details]
	Specified [Base /Range]:
	 (bytes) (MB)
	Total Memory (MB): 24448
	```

Output rx8640	`[Virtual Partition Details]`
	`Name: doiap17`
	`State: Up`
	`Attributes: Dynamic,Autoboot,Nosearch`
	`Kernel Path: /stand/vmunix`
	`Boot Opts: -lq`
	`[CPU Details]`
	`Min/Max: 1/12`
	`User assigned [Path]:`
	`Boot processor [Path]: 1.122`
	`Monitor assigned [Path]:`
	`Non-cell-specific:`
	`User assigned [Count]: 1`
	` Monitor assigned [Count]: 0`
	`Cell-specific [Count]: Cell ID/Count`
	` <none>`
	`[IO Details]`
	` 0.0.8`
	` 0.0.8.1.0.4.0.8.0.255.0.13.0`
	` 0.0.8.1.0.4.0.8.0.255.0.12.0 BOOT`
	` 0.0.8.1.0.4.1.8.0.255.0.13.0,ALTBOOT`
	`[Memory Details]`
	`ILM, user-assigned [Base /Range]:`
	` (bytes) (MB)`
	`ILM, monitor-assigned [Base /Range]: 0x11c0000000/8192`
	` (bytes) (MB)`
	`ILM Total (MB): 8192`
	`ILM Granularity (MB): 512`
	`CLM, user-assigned [CellID Base /Range]:`
	` (bytes) (MB)`
	`CLM, monitor-assigned [CellID Base /Range]:`

	`(bytes) (MB)`
	`CLM (CellID MB):`
	`CLM Granularity (MB): 128`	

Values taken	Const "HP vPar Config" > name	
	Name > vpar_name	
	Boot Opts > boot_options	
	Boot processor [Path] > boot_processor_ path	This value does not exist for rp8420 servers
	State > vpar_status	
	Attributes: Dynamic, Autoboot, Nosearch	• autoboot_mode: Autoboot • autosearch_mode: Nosearch • modification_mode: Dynamic
	Bound by User [Path]/User assigned [Path] > cpus_bound_by_user	Actual parameter is different between server versions
	Unbound [Path] > unbound_cpus	
Comment	For the attribute format of attributes such as **cpus_bound_by_user**, refer to the Data Model specification.	

Get Fibre Channel Adapters

Goal	To model Fibre Channel adapters
Command	ioscan -FnkCfc

Output	`pci:wsio:F:T:F:-1:50:4294967295:fc:fcd:`
	`0/0/12/1/0/4/0:16 119 35 18 0 0 0 0 :0:`
	`root.cell.sba.lba.PCItoPCI.fcd:fcd:`
	`CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X`
	` Fibre Channel 2-port 2Gb FC`
	`/2-port 1000B-T Combo Adapter (FC Port 1):0`
	
	` /dev/fcd0`
	
	`pci:wsio:F:T:F:-1:50:4294967295:fc:fcd:`
	`0/0/12/1/0/4/1:16 119 35 18 0 0 0 0 :1:`
	`root.cell.sba.lba.PCItoPCI.fcd:fcd:`
	`CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X Fibre Channel`
	`2-port 2Gb FC/2-port 1000B-T Combo Adapter (FC Port 2):1`
	
	` /dev/fcd1`
	
	`pci:wsio:F:T:F:-1:50:4294967295:fc:fcd:`
	`0/0/14/1/0/4/0:16 119 35 18 0 0 0 0 :`
	`2:root.cell.sba.lba.PCItoPCI.fcd:fcd:`
	`CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X Fibre Channel`
	`2-port 2Gb FC/2-port 1000B-T Combo Adapter (FC Port 1):2`
	
	` /dev/fcd2`
	
	`pci:wsio:F:T:F:-1:50:4294967295:fc:fcd:`
	`0/0/14/1/0/4/1:16 119 35 18 0 0 0 0 :3:`
	`root.cell.sba.lba.PCItoPCI.fcd:fcd:`
	`CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X`
	` Fibre Channel 2-port 2Gb FC/2-port`
	`1000B-T Combo Adapter (FC Port 2):3`
	
	` /dev/fcd3`	

Values taken	name	/dev/fcd0
	data_description	HP AB465-60001 PCI/PCI-X Fibre Channel 2-port 2Gb FC/2-port 1000B-T Combo Adapter (FC Port 2)
Comment	The hardware path serves to locate the Cell and use it as a container for FC HBA. Example value: `0/0/14/1/0/4/0`. The first integer value is the Global ID of the Cell; the second value is the ID of the I/O chassis.	

Get Disk Devices

Goal	To retrieve information about the dependency between I/O chassis, physical disk, and SCSI adapter.
Command	ioscan -FnkCdisk

Output	`scsi:wsio:T:T:F:31:188:2031616: disk:sdisk:0/0/12/1/0/4/0. 111.88.19.5.0.0:0 0 4 50 0 0 0 0 51 248 164 14 99 72 178 210 :3:root.cell.sba.lba.PCItoPCI.fcd.fcd_fcp.fcd_vbus.tgt.sdisk: sdisk:CLAIMED:DEVICE:EMC SYMMETRIX:31`
	`/dev/dsk/c31t0d0 /dev/rdsk/c31t0d0`
	`scsi:wsio:T:T:F:31:188:2031872: disk:sdisk:0/0/12/1/0/4/0. 111.88.19.5.0.1:0 0 4 50 0 0 0 0 51 248 164 14 76 238 217 30 :59:root.cell.sba.lba. PCItoPCI.fcd.fcd_fcp.fcd_vbus.tgt. sdisk:sdisk:CLAIMED:DEVICE:EMC SYMMETRIX:31`
	`/dev/dsk/c31t0d1 /dev/rdsk/c31t0d1`
	`scsi:wsio:T:T:F:31:188:2032128: disk:sdisk:0/0/12/1/0/4/0. 111.88.19.5.0.2:0 0 4 50 0 0 0 0 51 248 164 14 101 17 172 238 :61:root.cell.sba.lba. PCItoPCI.fcd.fcd_fcp.fcd_vbus.tgt.sdisk:sdisk: CLAIMED:DEVICE:EMC SYMMETRIX:31`
	`/dev/dsk/c31t0d2 /dev/rdsk/c31t0d2`	

Values taken	slot_number	0/0/12/1/0/4/0.111.88.19.5.0.0
	name	/dev/dsk/c31t0d2
	Cell ID	0/0/12/1/0/4/0.111.88.19.5.0.0
	IO chassis ID	0/0/12/1/0/4/0.111.88.19.5.0.0

Get Network Interfaces

Goal	To retrieve information about the dependency between network interfaces and the I/O chassis.
Command	ioscan -FnkClan

Output	`pci:wsio:F:F:F:-1:-1:4294967295:lan: igelan:0/0/12/1/0/6/0:20 228 22 72 0 0 0 0 :0: root.cell.sba.lba.PCItoPCI.igelan:igelan: CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X 1000Base-T 2-port 2Gb FC/2-port 1000B-T Combo Adapter:0`
	
	`pci:wsio:F:F:F:-1:-1:4294967295:lan: igelan:0/0/12/1/0/6/1:20 228 22 72 0 0 0 0 :1: root.cell.sba.lba.PCItoPCI.igelan:igelan: CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X 1000Base-T 2-port 2Gb FC/2-port 1000B-T Combo Adapter:1`
	
	`pci:wsio:F:F:F:-1:-1:4294967295:lan: igelan:0/0/14/1/0/6/0:20 228 22 72 0 0 0 0 :2: root.cell.sba.lba.PCItoPCI.igelan:igelan: CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X 1000Base-T 2-port 2Gb FC/2-port 1000B-T Combo Adapter:2`
	
	`pci:wsio:F:F:F:-1:-1:4294967295:lan: igelan:0/0/14/1/0/6/1:20 228 22 72 0 0 0 0 :3: root.cell.sba.lba.PCItoPCI.igelan:igelan: CLAIMED:INTERFACE:HP AB465-60001 PCI/PCI-X 1000Base-T 2-port 2Gb FC/2-port 1000B-T Combo Adapter:3`

Values taken	The hardware path which reflects the Cell and I/O chassis that this interface belongs to.

Get File Systems

Goal	To retrieve information about the file systems and corresponding logical volumes.
Command	df -P

Output	`Filesystem 512-blocks Used Available Capacity Mounted on`
	
	`/dev/vg01/lv106 9837710 115094 9722616 2% /usr/vw/rvs`
	
	`/dev/vg01/lv124 7915344 814616 7100728 11% /home/kdov12`
	
	`/dev/vg01/lv125 10222640 6275190 3947450 62% /home/ebrev`
	
	`/dev/vg01/lv123 20829536 2796208 18033328 14% /home/temp`
	
	`/dev/vg01/lv110 2080832 4608 2076224 1% /oracle2/arch/inst_aebp`

Values taken	name for FileSystem CIT: /usr/vw/rvs Name of the logical volume: /dev/vg01/lv106

Get Logical Volumes, Volume Groups, and Physical Volumes

Goal	To retrieve data for modeling Logical volumes, Volume groups, and Physical volumes.
Command	vgdisplay -v

Output

```
--- Volume groups ---
VG Name /dev/vg00
VG Write Access read/write
VG Status available
Max LV 255
Cur LV 10
Open LV 10
Max PV 16
Cur PV 1
Act PV 1
Max PE per PV 4384
VGDA 2
PE Size (Mbytes) 16
Total PE 4315
Alloc PE 4156
Free PE 159
Total PVG 0
Total Spare PVs 0
Total Spare PVs in use 0
 --- Logical volumes ---
 LV Name /dev/vg00/lvol1
 LV Status available/syncd
 LV Size (Mbytes) 256
 Current LE 16
 Allocated PE 16
 Used PV 1
 --- Physical volumes ---
 PV Name /dev/dsk/c31t0d0
 PV Name /dev/dsk/c32t0d0 Alternate Link
 PV Status available
 Total PE 4315
```

	`Free PE 159`
	`Autoswitch On`
	`Proactive Polling On`	

Values taken	Volume group	
	VG Name > name VG Write Access > write_access	
	VG Status > vg_status PE Size (Mbytes)	This value is used to calculate the size of the physical volume
	Logical Volume	
	LV Name > name LV Status > lv_status	
	Physical Volume	
	PV Name > name	Alternate link may also be used. It depends on the output of the **ioscan FnkCdisk** command.
	PV Status > pv_status	
	Total PE > pv_size	This attribute is calculated on the PE Size (Mbytes) value.

Get Network Interfaces

Goal	To retrieve information about the network interfaces.
Command	lanscan

Output	`Hardware Station Crd Hdw Net-Interface NM MAC HP-DLPI DLPI`
	`Path Address In# State NamePPA ID Type Support Mjr#`
	`0/0/4/1/0/6/1 0x0014C254D9BD 1 UP lan1 snap1 2 ETHER Yes 119`
	`0/0/6/1/0/6/1 0x0014C254C961 3 UP lan3 snap3 4 ETHER Yes 119`
	`LinkAgg0 0x0014C254D9BC 900 UP lan900 snap900 6 ETHER Yes 119`
	`LinkAgg1 0x000000000000 901 DOWN lan901 snap901 7 ETHER Yes 119`
	`LinkAgg2 0x000000000000 902 DOWN lan902 snap902 8 ETHER Yes 119`
	`LinkAgg3 0x000000000000 903 DOWN lan903 snap903 9 ETHER Yes 119`
	`LinkAgg4 0x000000000000 904 DOWN lan904 snap904 10 ETHER Yes 119`

Values taken	• The hardware path to create the link between the network interface and I/O chassis. • The MAC address to create the network interface. • The MAC address of the Link aggregation interface, the indicator that the interface is up, and the device name.

Get Information About Link Aggregation Interfaces

Goal	To model the links between interfaces and link aggregation.
Command	lanscan -q

Output	`1`
	`3`
	`900 0 2`
	`901`
	`902`
	`903`
	`904`

Values taken	The interface number and IDs of the aggregated interfaces.

Get MAC Addresses of the Aggregated Interfaces

Goal	To retrieve the MAC addresses of the aggregated interfaces.

Command	lanadmin -a <interface_id>
Example	lanscan -a 0
Output	`Station Address = 0x0014c254d9bc`
Values taken	The MAC address of the aggregated interface

Get Hardware Paths of the Aggregated Interfaces

Goal	To retrieve the hardware path of the aggregated interfaces
Command	lanscan -v | grep -E <list_of_aggregated_interfaces>
Example	lanscan -v | grep -E "lan0|lan2"

Output	`0/0/4/1/0/6/0 0 UP lan0 snap0 1 ETHER Yes 119 igelan`
	`0/0/6/1/0/6/0 2 UP lan2 snap2 3 ETHER Yes 119 igelan`

Values taken	The hardware path that allocates the I/O chassis that holds this interface.

Get IP Addresses of the Aggregated Interfaces

Goal	To get IP addresses of the interfaces
Command	netstat -rn

Output	`Routing tables`
	`Destination Gateway Flags Refs Interface Pmtu`
	`127.0.0.1 127.0.0.1 UH 0 lo0 4136`
	`10.186.112.115 10.186.112.115 UH 0 lan0 4136`
	`10.186.116.13 10.186.116.13 UH 0 lan1 4136`
	`192.168.121.1 192.168.121.1 UH 0 lan2 4136`
	`10.186.115.18 10.186.115.18 UH 0 lan3 4136`
	`10.186.116.19 10.186.116.19 UH 0 lan1:1 4136`
	`10.186.116.0 10.186.116.13 U 3 lan1 1500`
	`10.186.116.0 10.186.116.19 U 3 lan1:1 1500`
	`10.186.115.0 10.186.115.18 U 2 lan3 1500`
	`10.186.112.0 10.186.112.115 U 2 lan0 1500`
	`192.168.121.0 192.168.121.1 U 2 lan2 1500`
	`10.186.86.0 10.186.115.1 UG 0 lan3 1500`
	`127.0.0.0 127.0.0.1 U 0 lo0 4136`
	`default 10.186.116.1 UG 0 lan1 1500`

Values taken	The IP addresses of the interfaces. The **netstat** command does not require root privileges, in contrast to **ifconfig**.

Troubleshooting and Limitations

- **The destination host is not a part of the HP nPartition system.**

  DFM considers the target host as not being a part of the HP partitionable system. The criteria are based on executing the **parstatus -s** command.

- **Failed to discover vPartition details.**

  The **vparstatus** command was not executed successfully. This command should be accessible and DFM should have enough permissions to execute it. If this command requires **sudo** to be executed, configure the SSH credentials.

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

- **Failed to discover storage topology.**

  The **vgdisplay** command was not executed successfully.

- **Failed to link file systems and disks.**

  The **df** command was not executed successfully.

- **Failed to discover SCSI adapters, or Fibre Channel adapters, or Network cards.**

  The **ioscan** command was not executed successfully.

Chapter 10: Hyper-V Discovery

This chapter includes:

Overview

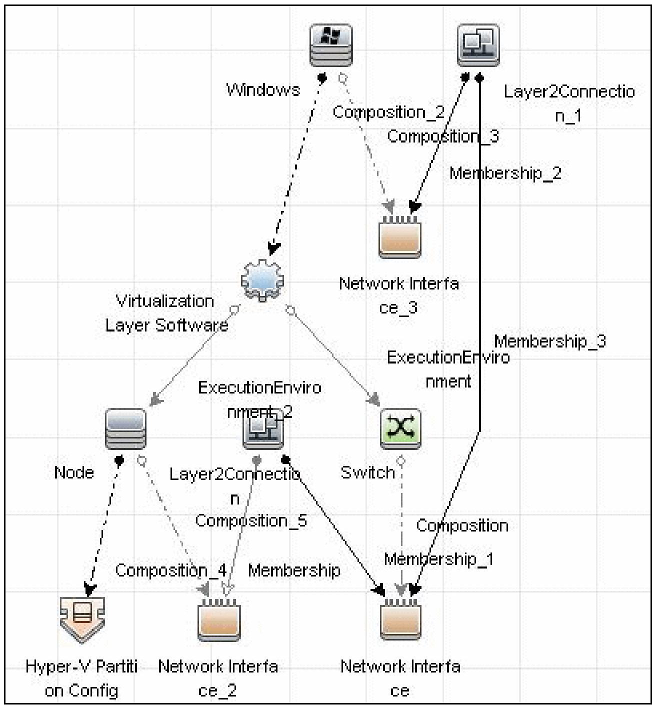
The **Hyper-V** package discovers the Hyper-V Aware Windows server through WMI and NTCMD. It discovers resource pools, virtual switches, virtual NICs, and virtual machines.

Supported Versions

The **Hyper-V** package supports Windows 2008, Windows 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Topology

The following image displays the topology of the Hyper-V discovery:



How to Discover Hyper-V

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the NTCMD and WMI protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Verification**

   Verify that you can perform WMI queries in the **\\root\virtualization** namespace or the **\\root\virtualization/v2** namespace on the target machine, either through WMI or through the **wmic** command when connecting through a Shell protocol.

3. **Run the Discovery**

   To discover Hyper-V topology through Shell:

   a. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   b. Run the **Host Connection by Shell** job to discover Shell connectivity and basic information about the hosts.

   c. Run the **Host Applications by Shell** job to discover processes on target machines.

   d. Run the **Hyper-V Topology by Shell** job to discover the Hyper-V topology.

   To discover Hyper-V topology through WMI:

   a. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   b. Run the **Host Connection by WMI** job to discover WMI connectivity and basic information about the hosts.

   c. Run the **Host Applications by WMI** job to discover processes on target machines.

   d. Run the **Hyper-V Topology by WMI** job to discover Hyper-V topology.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Discovery Mechanism

Contains the enumeration of WMI classes and attributes for supported namespaces.

Discovery Mechanism for \\root\virtualization Namespace

This section includes the following commands:

Retrieve the Hyper-V Host Name

Object queried	Msvm_ComputerSystem
Conditions	Description = 'Microsoft Hosting Computer System'
Properties queried	ElementName

Comments	Verifies that the Hyper-V namespace **\\root\virtualization** is accessible and obtains the name of the Hyper-V host.

Retrieve the Virtual Machine

Object queried	Msvm_ComputerSystem
Conditions	Description = 'Microsoft Virtual Machine'
Properties queried	• Name • ElementName • EnabledState • HealthState
Comments	Obtains virtual machines present in the Hyper-V host, and obtains GUID, name health, and enabled states for each virtual machine.

Retrieve the Global Settings for Virtual Machines

Object queried	Msvm_VirtualSystemGlobalSettingData
Conditions	None
Properties queried	• SystemName • SnapshotDataRoot • ExternalDataRoot • AutomaticRecoveryAction • AutomaticShutdownAction • AutomaticStartupAction
Comments	Obtains global settings for all virtual machines.

Retrieve the Settings for Virtual Machines

Object queried	Msvm_VirtualSystemSettingData

Conditions	None
Properties queried	• InstanceID • BaseBoardSerialNumber • BIOSGUID • BIOSSerialNumber • ChassisAssetTag • ChassisSerialNumber
Comments	Obtains the **VirtualSystemSettingData** (VSSD) objects of the virtual machines that hold additional settings for virtual machines. The **BIOSGUID** property holds the BIOS UUID of the virtual machine. This property is stripped of leading and trailing curly brackets ({}).

Retrieve the References from Virtual Machines to Settings (VSSD)

Object queried	Msvm_SettingsDefineState
Conditions	None
Properties queried	• ManagedElement • SettingData
Comments	Associates virtual machines and their settings (**VirtualSystemSettingData**).

Retrieve the References from Virtual Machine Settings (VSSD) to Components

Object queried	Msvm_VirtualSystemSettingDataComponent
Conditions	None
Properties queried	• GroupComponent • PartComponent
Comments	Obtains references from the **VirtualSystemSettingData** object to its components.

Retrieve the Memory Settings for Virtual Machines

Object queried	Msvm_MemorySettingData
Conditions	None
Properties queried	• InstanceID • Limit • Reservation
Comments	Obtains memory settings for virtual machines (reservation and limit). The references retrieved during the previous step ("Discovery Mechanism" on page 117) enable the correct association of these settings to the relevant virtual machine.

Retrieve the Processor Settings for Virtual Machines

Object queried	Msvm_ProcessorSettingData
Conditions	None
Properties queried	• InstanceID • Limit • Reservation • Weight
Comments	Obtains processor settings for virtual machines (reservation, limit, weight). The references retrieved during a previous step ("Discovery Mechanism" on page 117) enable the correct association of these settings to the relevant virtual machine.

Retrieve Virtual Switches

Object queried	Msvm_VirtualSwitch
Conditions	None
Properties queried	• ElementName • Name

Comments	Obtains virtual switches configured on a Hyper-V host.

Retrieve the Ports of Virtual Switches

Object queried	Msvm_SwitchPort
Conditions	None
Properties queried	• ElementName • Name
Comments	Obtains the ports on virtual switches.

Retrieve the References from Virtual Switches to Ports

Object queried	Msvm_HostedAccessPoint
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references that enable associating virtual switches and their ports.

Retrieve the Interfaces of Virtual Machines

Object queried	Msvm_VmLANEndpoint
Conditions	None
Properties queried	• Name • ElementName • MACAddress
Comments	Obtains endpoints that are connected to interfaces of virtual machines. Although these endpoints are not interfaces themselves, they hold enough information to report interfaces.

Retrieve the Interfaces of Management Partitions

Object queried	Msvm_SwitchLANEndpoint
Conditions	None
Properties queried	• Name • ElementName • MACAddress
Comments	Obtains endpoints that are connected to interfaces of a Management Partition (on a Hyper-V host). Although these endpoints are not interfaces themselves, they hold enough information to report interfaces. They include both physical interfaces and virtual interfaces of the partition used for internal connections to virtual machines.

Retrieve the References from Virtual Machines to Interfaces

Object queried	Msvm_DeviceSAPImplementation
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references from virtual endpoints to virtual machines, thus enabling associations.

Retrieve the References from Ports on Virtual Switches to Interfaces

Object queried	Msvm_ActiveConnection
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references from a port on a virtual switch to endpoints that enable associations.

Discovery Mechanism for \\root\virtualization\v2 Namespace

This section includes the following commands:

Retrieve the Hyper-V Host Name

Object queried	Msvm_ComputerSystem
Conditions	Description = 'Microsoft Hosting Computer System'
Properties queried	ElementName
Comments	Verifies that the Hyper-V namespace **\\root\virtualization\v2** is accessible and obtains the name of the Hyper-V host.

Retrieve the Virtual Machine

Object queried	Msvm_ComputerSystem
Conditions	Description = 'Microsoft Virtual Machine'
Properties queried	• Name • ElementName • EnabledState • HealthState
Comments	Obtains virtual machines present in the Hyper-V host, and obtains GUID, name health, and enabled states for each virtual machine.

Retrieve the Global Settings for Virtual Machines

Object queried	Msvm_VirtualSystemSettingData
Conditions	None

Properties queried	• InstanceID • SnapshotDataRoot • ExternalDataRoot • AutomaticRecoveryAction • AutomaticShutdownAction • AutomaticStartupAction
Comments	Obtains global settings for all virtual machines.

Retrieve the Settings for Virtual Machines

Object queried	Msvm_VirtualSystemSettingData
Conditions	None
Properties queried	• InstanceID • BaseBoardSerialNumber • BIOSGUID • BIOSSerialNumber • ChassisAssetTag • ChassisSerialNumber
Comments	Obtains the **VirtualSystemSettingData** (VSSD) objects of the virtual machines that hold additional settings for virtual machines. The **BIOSGUID** property holds the BIOS UUID of the virtual machine. This property is stripped of leading and trailing curly brackets ({}).

Retrieve the References from Virtual Machines to Settings (VSSD)

Object queried	Msvm_SettingsDefineState
Conditions	None

Properties queried	• ManagedElement • SettingData
Comments	Associates virtual machines and their settings (**VirtualSystemSettingData**).

Retrieve the References from Virtual Machine Settings (VSSD) to Components

Object queried	Msvm_VirtualSystemSettingDataComponent
Conditions	None
Properties queried	• GroupComponent • PartComponent
Comments	Obtains references from the **VirtualSystemSettingData** object to its components.

Retrieve the Memory Settings for Virtual Machines

Object queried	Msvm_MemorySettingData
Conditions	None
Properties queried	• InstanceID • Limit • Reservation
Comments	Obtains memory settings for virtual machines (reservation and limit). The references retrieved during the previous step ("Discovery Mechanism" on page 117) enable the correct association of these settings to the relevant virtual machine.

Retrieve the Processor Settings for Virtual Machines

Object queried	Msvm_ProcessorSettingData
Conditions	None

Properties queried	• InstanceID • Limit • Reservation • Weight
Comments	Obtains processor settings for virtual machines (reservation, limit, weight). The references retrieved during a previous step ("Discovery Mechanism" on page 117) enable the correct association of these settings to the relevant virtual machine.

Retrieve Virtual Switches

Object queried	Msvm_VirtualEthernetSwitch
Conditions	None
Properties queried	• ElementName • Name
Comments	Obtains virtual switches configured on a Hyper-V host.

Retrieve the Ports of Virtual Switches

Object queried	Msvm_EthernetSwitchPort
Conditions	None
Properties queried	• ElementName • Name
Comments	Obtains the ports on virtual switches.

Retrieve the Interfaces of Virtual Machines

Object queried	Msvm_LANEndpoint
Conditions	None

Properties queried	• Name • ElementName • MACAddress
Comments	Obtains endpoints that are connected to interfaces of virtual machines. Although these endpoints are not interfaces themselves, they hold enough information to report interfaces.

Retrieve the References from Virtual Machines to Interfaces

Object queried	Msvm_DeviceSAPImplementation
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references from virtual endpoints to virtual machines, thus enabling associations.

Retrieve the References from Ports on Virtual Switches to Interfaces

Object queried	Msvm_ActiveConnection
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references from a port on a virtual switch to endpoints that enable associations.

Retrieve the Interfaces of Hyper-V Host

Object queried	Msvm_LANEndpoint
Conditions	None

Properties queried	• Name • ElementName • MACAddress
Comments	Obtains the interfaces of a Hyper-V host.

Retrieve the Synthetic Ethernet Adapter

Object queried	Msvm_SyntheticEthernetPort
Conditions	None
Properties queried	• DeviceID • ElementName • PermanentAddress • SystemName
Comments	Obtains the synthetic Ethernet interfaces.

Retrieve the Emulated Ethernet Adapter

Object queried	Msvm_EmulatedEthernetPort
Conditions	None
Properties queried	• DeviceID • ElementName • PermanentAddress • SystemName
Comments	Obtains the emulated Ethernet interfaces.

Retrieve the Internal Ethernet Adapter (network adapter)

Object queried	Msvm_InternalEthernetPort

Conditions	None
Properties queried	• DeviceID • ElementName • PermanentAddress • SystemName
Comments	Obtains the internal Ethernet interfaces.

Retrieve the External Ethernet Adapter (network adapter)

Object queried	Msvm_ExternalEthernetPort
Conditions	None
Properties queried	• DeviceID • ElementName • PermanentAddress • SystemName
Comments	Obtains the external Ethernet interfaces.

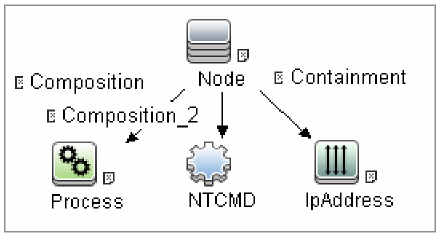
Retrieve the References from LAN Endpoints to a Global Ethernet Port

Object queried	Msvm_EthernetDeviceSAPImplementation
Conditions	None
Properties queried	• Antecedent • Dependent
Comments	Obtains references from LAN endpoints to a global Ethernet port.

Hyper-V Topology by Shell Job

This section includes information about the trigger query and adapter for this job.

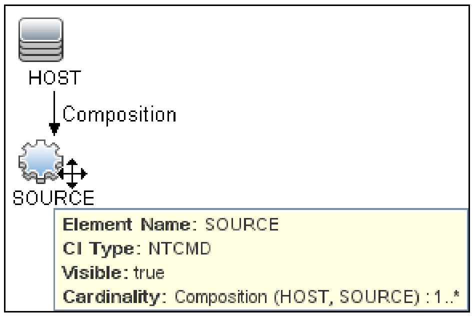
Trigger Query



Adapter

This job uses the **hyperv_topology_by_shell** adapter.

- Input Query



Element Name: SOURCE
CI Type: NTCMD
Visible: true
Cardinality: Composition (HOST, SOURCE) : 1..*

- CI Attribute Conditions

Attribute	Condition
Process	Name Equal ignore case "vmms.exe"
NTCMD	NOT Reference to the credentials dictionary entry Is null
IpAddress	NOT IP Probe Name Is null

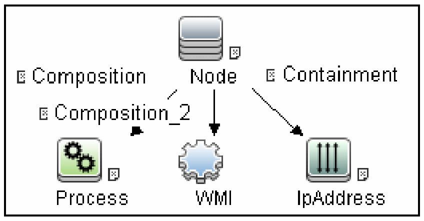
- Discovered CITs
  - Composition
  - ExecutionEnvinroment
  - Hyper-V Partition Config

- Interface
- Layer2Connection
- Membership
- Node
- Switch
- Virtualization Layer Software

Hyper-V Topology by WMI Job

This section includes information about the trigger query and adapter for this job.

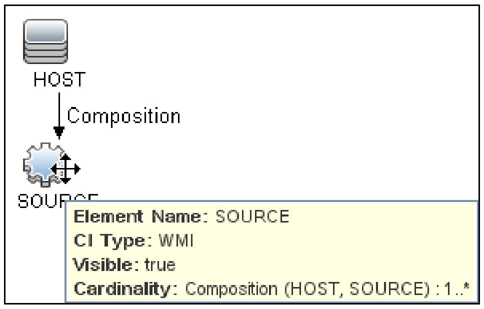
Trigger Query



Adapter

This job uses the **hyperv_topology_by_wmi** adapter.

- Input Query



Element Name: SOURCE
CI Type: WMI
Visible: true
Cardinality: Composition (HOST, SOURCE) : 1..*

- CI Attribute Conditions

Attribute	Condition
Process	Name Equal ignore case "vmms.exe"
WMI	NOT Reference to the credentials dictionary entry Is null
IpAddress	NOT IP Probe Name Is null

- Discovered CITs
  - Composition
  - ExecutionEnvinroment
  - Hyper-V Partition Config
  - Interface
  - Layer2Connection
  - Membership
  - Node
  - Switch
  - Virtualization Layer Software

Created/Changed Entities

Entity	**New/Changed**	**Entity Name**
CITs	New	Hyper-V Partition Config (hyperv_partition_config)
Valid links	New	None
Views	New	Hyper-V Topology
Scripts	New	• hyperv_topology_by_shell.py • hyperv_topology_by_wmi.py • hyperv.py
Adapters	New	• hyperv_topology_by_shell • hyperv_topology_by_wmi
Jobs	New	• Hyper-V Topology by Shell • Hyper-V Topology by WMI
Trigger Queries		• ntcmd_on_hyperv_host • wmi_on_hyperv_host

Troubleshooting and Limitations

Virtual machines that are offline cannot be discovered, since the information about their MAC address is not available.

Chapter 11: IBM Virtualization Discovery

This chapter includes:

Overview

This chapter describes the usage and functionality of the IBM Virtualization discovery package. This package supports discovery of IBM Virtualization Topology based on one of the IBM virtualization managers Hardware Management Console (HMC) or Integrity Virtual Machines (IVM) as data sources.

IBM HMC was invented by IBM for the purpose of providing a standard interface for configuring and operating partitioned (also known as an LPAR or virtualized system) and SMP systems such as IBM System I or IBM System p series.

IBM IVM is an easy-to-use, browser-based tool that allows clients to point, click, and consolidate multiple workloads into one IBM Power System.

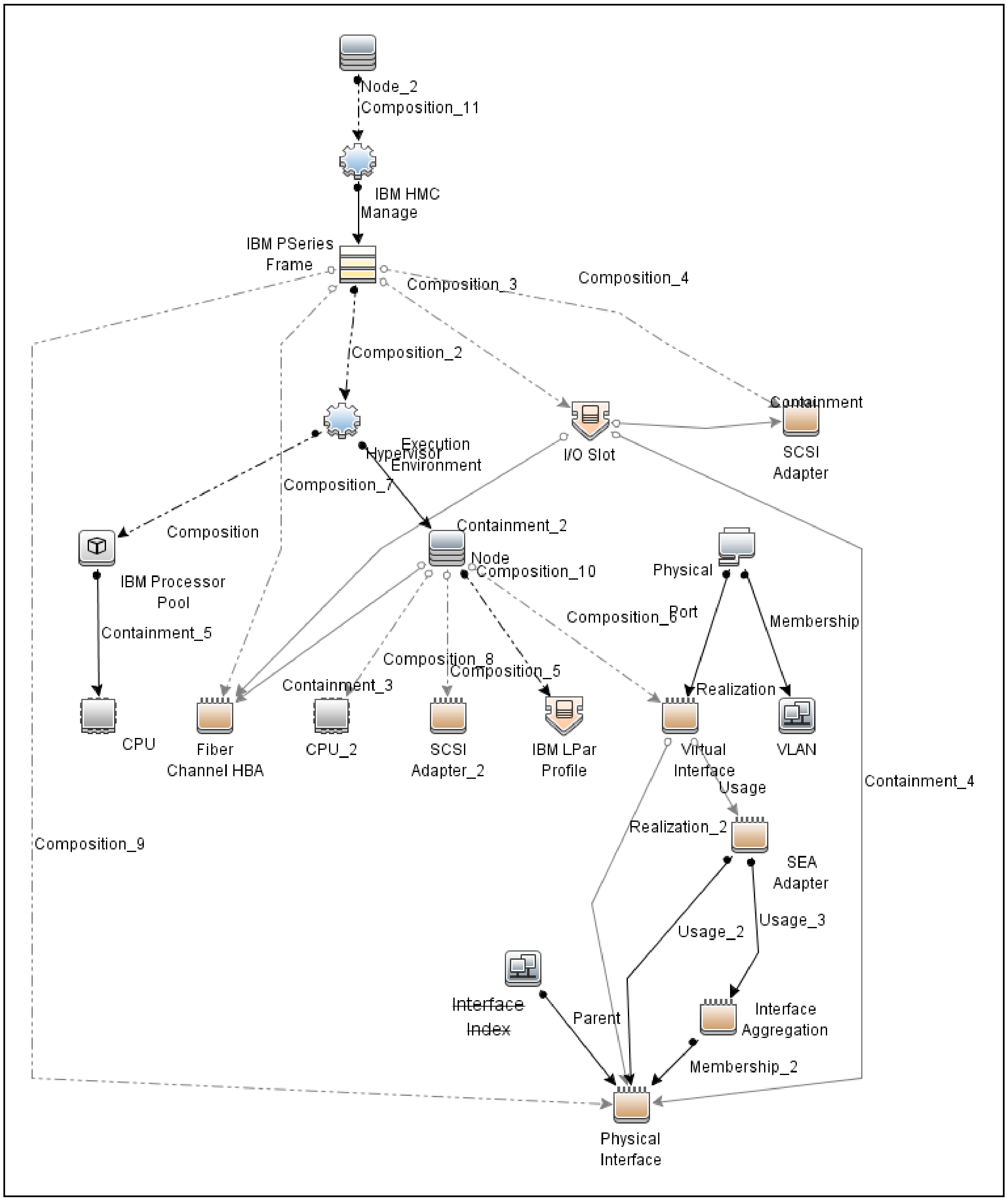
Supported Versions

This discovery solution supports IBM HMC versions 3.x, 5.x, 6.x and 7.x on AIX and Linux.

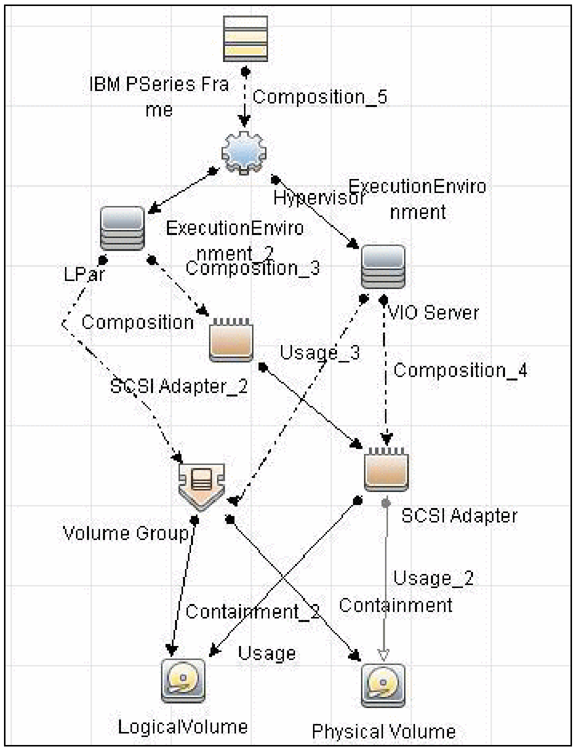
Topology

Note: For a list of discovered CITs, see "IBM LPar and VIO Server Topology by Shell Job" on page 148.

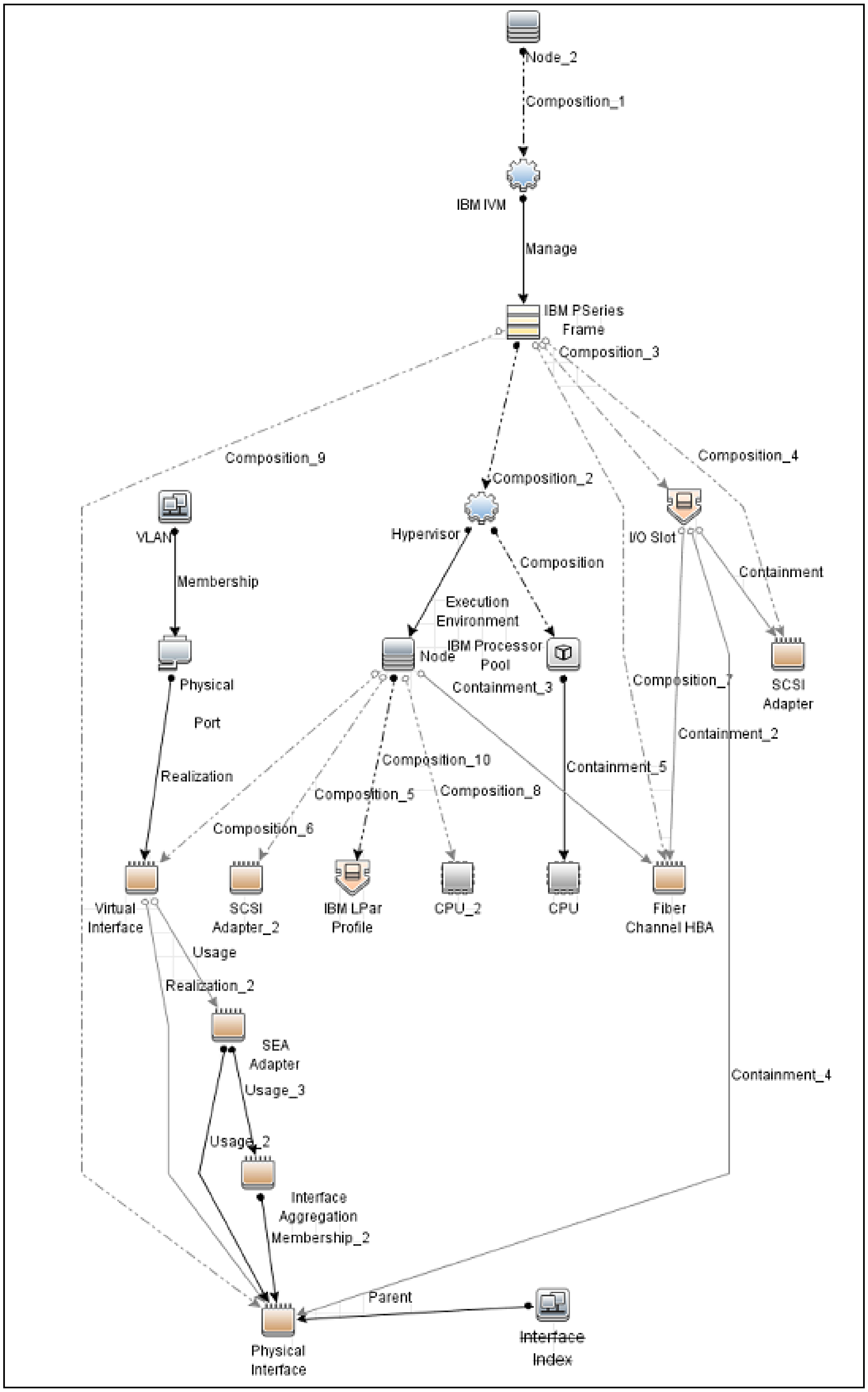
IBM Virtualization by Shell Topology



IBM Storage Topology



IBM IVM Topology


Node_2
Composition_1
IBM IVM
Manage
IBM PSeries Frame
Composition_3
Composition_9
Composition_2
Composition_4
VLAN
Hypervisor
I/O Slot
Membership
Composition
Containment
Execution Environment
Physical Port
Node
IBM Processor Pool
Containment_3
Composition_7
SCSI Adapter
Containment_2
Realization
Composition_10
Containment_5
Composition_5
Composition_8
Composition_6
Virtual Interface
SCSI Adapter_2
IBM LPar Profile
CPU_2
CPU
Fiber Channel HBA
Usage
Realization_2
SEA Adapter
Usage_3
Containment_4
Usage_2
Interface Aggregation
Membership_2
Parent
Physical Interface
Interface Index

How to Discover IBM Virtualization

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the SSH and Telnet Shell protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   If some of the commands are configured to run with **sudo** on the target host, in the **Protocol Parameters** dialog box, fill in the following fields:

   - **Sudo paths**. Enter the full path to the sudo executable, together with the name of the executable. You can add more than one entry if executable files are placed in various places on the target operating systems.

     Example: `sudo,/usr/bin/sudo,/bin/sudo`

   - **Sudo commands**. Enter a list of commands that are prefixed with **sudo**.

     Example: `lspath,ifconfig`

     For details, see "Protocol Parameter Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

2. **Prerequisites - Set up permissions**

   Before activating discovery, confirm that the discovery user has all the required permissions to run the following commands.

   Note: For details about these commands, see:

   - "IBM Virtualization Commands" on page 152
   - "VIO Server Side Commands" on page 164
   - "LPAR Side Commands" on page 174

- hostname
- lscfg
- lsdev -dev <Device>
- lshmc -b
- lshmc -n
- lshmc -v
- lshmc -V
- lshwres -r io --rsubtype slot -m <pSeriesName>
- lshwres -r mem --level lpar -m <pSeriesName>
- lshwres -r mem --level sys -m <pSeriesName>
- lshwres -r proc --level lpar -m <pSeriesName>
- lshwres -r proc --level pool -m <pSeriesName>
- lshwres -r proc --level sys -m <pSeriesName>
- lshwres -r virtualio --rsubtype eth --level lpar -m <pSeriesName>
- lshwres -r virtualio --rsubtype scsi -m <pSeriesName>
- lsivm
- lslv
- lslv -v <Logical Volume Name>
- lsmap -all
- lsmap -all -net
- lspartition
- lspath

- lspv
- lssyscfg -r lpar -m <pSeriesName>
- lssyscfg -r prof -m <pSeriesName> --filter <lparName>
- lssyscfg -r sys
- lstcpip
- lsvg
- lsvg -l <Volume Group Name>
- lsvio -e
- lsvio -s
- lvdisplay
- pvdisplay
- vgdisplay

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job.

   b. Run the **Host Connection by Shell** job.

   c. Run the **IBM Virtualization by Shell** job.

   d. Run the **IBM LPar and VIO by Shell** job.

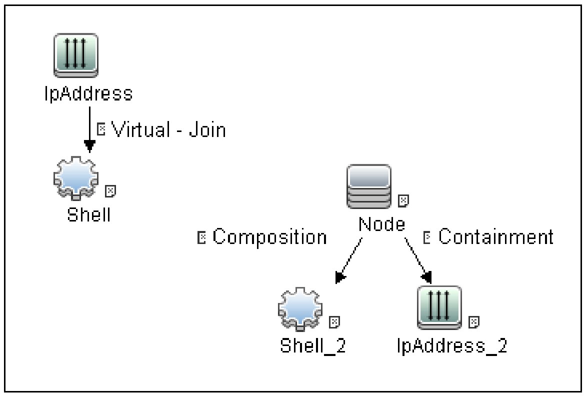
   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

IBM Virtualization by Shell Job

This section includes:

- "IBM Virtualization by Shell Job" on the previous page
- "IBM Virtualization by Shell Job" on the previous page
- "IBM Virtualization by Shell Job" on the previous page

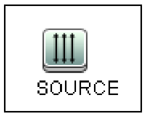
Trigger Query



Adapter

This job uses the **IBM_VIRTUALIZATION_SHELL_PATTERN** adapter.

- Input Query



- Triggered CI Data

Name	Value
ip_address	${SOURCE.ip_address}
ip_domain	${SOURCE.ip_domain}

- Used Scripts

- ibm_hmc_by_shell.py
- storage_topology.py
- ibm_hmc_lib.py

Discovered CITs

- Composition
- Containment
- Cpu
- ExecutionEnvironment
- I/O Slot
- IBM Frame
- IBM HMC
- IBM IVM
- IBM LPar Profile
- IBM Processor Pool
- Interface
- IpAddress
- Manage
- Membership
- Node
- PhysicalPort
- Realization

- SCSI Adapter
- Shell
- Usage
- Virtualization Layer Software
- Vlan

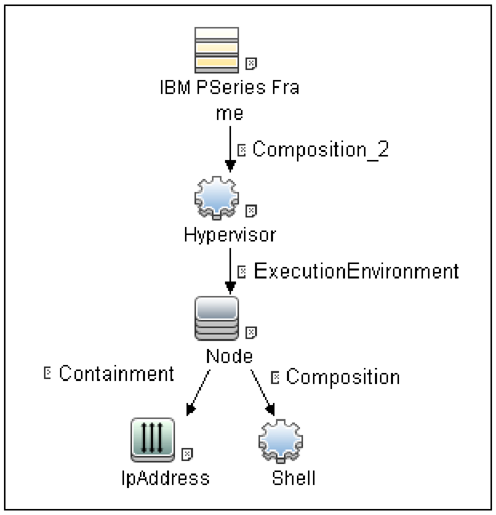
Note: To view the topology, see "IBM Virtualization by Shell Topology" on page 139.

IBM LPar and VIO Server Topology by Shell Job

This section includes:

- "IBM LPar and VIO Server Topology by Shell Job" above
- "IBM LPar and VIO Server Topology by Shell Job" above
- "IBM LPar and VIO Server Topology by Shell Job" above

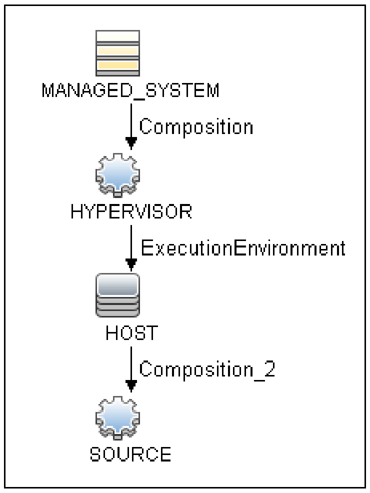
Trigger Query



Adapter

This job uses the **IBM_LPAR_VIO_BY_SHELL** adapter.

- Input Query



- Triggered CI Data

Name	**Value**
Protocol	${SOURCE.root_class}
credentialsId	${SOURCE.credentials_id}
hostId	${SOURCE.root_container}
ip_address	${SOURCE.application_ip}
managedSystemId	${MANAGED_SYSTEM.root_id}
osType	${HOST.host_os}

- Used Scripts
  - ibm_hmc_lib.py
  - ibm_lpar_or_vio_by_shell.py
  - storage_topology.py

Discovered CITs

- Composition
- Containment
- Dependency
- Fibre Channel HBA
- FileSystem
- I/O Slot
- Interface
- Interface Aggregation
- Interface Index
- IpAddress
- IVM
- LogicalVolume
- Membership
- Node
- Parent
- Physical Volume
- Realization
- SCSI Adapter
- SEA Adapter
- Usage
- Volume Group

Note: To view the topology, see "Topology" on page 138.

IBM Virtualization Commands

This section includes the following commands:

lshmc -V

Output

```
version= Version: 7 Release: 3.5.0 Service Pack: 0 HMC Build level 20091201.1
MH01195: Required fix for HMC V7R3.5.0 (10-16-2009) MH01197: Fix for HMC V7R3.5.0
(11-12-2009) MH01204: Fix for HMC V7R3.5.0 (12-11-2009) ","base_version=V7R3.5.0 "
```

Mapping

The output of this command is used to fill in the attributes of the **IBM HMC** CI:

CMD Output Attribute	CI Name	CI Attribute
Version	IBM HMC	Version_number
Base_version	IBM HMC	Application_version_description

lshmc -v

Output

```
vpd=*FC ???????? *VC 20.0 *N2 Tue Apr 27 13:05:33 CEST 2010 *FC ???????? *DS
Hardware Management Console *TM eserver xSeries 335 -[XXXXCR2]- *SE XXXXXXX *MN IBM
*PN Unknown *SZ 1059495936 *OS Embedded Operating Systems *NA 192.168.1.10 *FC
???????? *DS Platform Firmware *RM V7R3.5.0.0
```

Mapping

The output of this command is used to fill in the attributes of the **IBM HMC** CI:

CMD Output Attribute	CI Name	CI Attribute
SE	IBM HMC	HMC Serial Number
TM	IBM HMC	HMC TYPE

lshmc -b

Output

```
bios=T2E139AUS-1.15
```

Mapping

The output of this command is used to fill in the attributes of the **IBM HMC** CI:

CMD Output Attribute	CI Name	CI Attribute
Bios	IBM HMC	HMC BIOS

lshmc -n

Output

```
hostname=hmc01,domain=somedomain.com,
"ipaddr=192.168.1.10,0.0.0.0,192.168.128.1",
"networkmask=255.255.254.0,255.255.255.0,255.255.128.0",
gateway=192.168.1.1,nameserver=,domainsuffix=,
slipipaddr=192.168.1.1,slipnetmask=255.255.0.0,
"ipaddrlpar=192.168.80.1,192.168.128.1",
"networkmasklpar=255.255.254.0,255.255.128.0",
clients=,ipv6addrlpar=,ipv4addr_eth0=192.168.1.10,
ipv4netmask_eth0=255.255.254.0,ipv4dhcp_eth0=off,ipv6addr_eth0=,
ipv6auto_eth0=off,ipv6privacy_eth0=off,ipv6dhcp_eth0=off,
lparcomm_eth0=off,jumboframe_eth0=off,speed_eth0=100,
duplex_eth0=full,tso_eth0=off,ipv4addr_eth1=0.0.0.0,
ipv4netmask_eth1=255.255.255.0,ipv4dhcp_eth1=off,
ipv6addr_eth1=,ipv6auto_eth1=off,ipv6privacy_
eth1=off,ipv6dhcp_eth1=off,lparcomm_eth1=off,jumboframe_
eth1=off,speed_eth1=auto,duplex_eth1=auto,tso_
eth1=off,ipv4addr_eth2=192.168.128.1,ipv4netmask_
eth2=255.255.128.0,ipv4dhcp_eth2=off,ipv6addr_
eth2=,ipv6auto_eth2=off,ipv6privacy_eth2=off,ipv6dhcp_
eth2=off,lparcomm_eth2=off,jumboframe_eth2=off,speed_
eth2=auto,duplex_eth2=auto,tso_eth2=off
```

Mapping

The output of this command is used to fill in the network information for a particular HMC machine. A host with HMC running on it is always reported as an incomplete host, since there is no information regarding the interface MAC addresses and the default UNIX command does not work in this environment.

CMD Output Attribute	CI Name	CI Attribute
constant AIX	Unix	Host Operating System
Hostname	Unix	Host Name
Hostname	Unix	Name
Domain	Unix	OS Domain Name
Ipv4addr_eth<0..N>	IpAddress	Ip Address

lspartition -c <TYPE>_<VERSION> -i

Output

```
2,192.168.80.52,3;1,192.168.80.62,3;3,192.168.80.53,3
```

Mapping

Each block in the output is separated by the semicolon character (;). The first value is the LPAR ID and the second value is the LPAR IP address. By matching the ID of the LPAR with output from other commands an incomplete host is created and reported with an assigned LPAR Profile CI.

lssyscfg -r sys

Output

```
name=XXXXXXX-XXXX-XXX-XXXXXXXXX-XX,type_model=XXXX-XXX, serial_
num=XXXXXX,ipaddr=192.168.1,10,state=Operating,sys_time=04/27/2010 12:55:23,power_
off_policy=1,active_lpar_mobility_capable=0,inactive_lpar_mobility_
capable=0,active_lpar_share_idle_procs_capable=0,active_mem_sharing_capable=0,bsr_
capable=0,cod_mem_capable=0,cod_proc_capable=1,electronic_err_reporting_
capable=0,firmware_power_saver_capable=0,hardware_power_saver_capable=0,hardware_
discovery_capable=0,addr_broadcast_perf_policy_capable=0,hca_capable=1,huge_page_
mem_capable=1,lhea_capable=0,lpar_avail_priority_capable=0,lpar_proc_compat_mode_
capable=0,micro_lpar_capable=1,os400_capable=0,5250_application_
capable=0,redundant_err_path_reporting_capable=1,shared_eth_failover_capable=1,sni_
msg_passing_capable=0,sp_failover_capable=1,vet_activation_capable=1,virtual_fc_
capable=0,virtual_io_server_capable=1,virtual_switch_capable=0,assign_5250_cpw_
percent=0,max_lpars=40,max_power_ctrl_lpars=1,hca_bandwidth_
capabilities=null,service_lpar_id=none,curr_sys_keylock=norm,pend_sys_
keylock=norm,curr_power_on_side=temp,pend_power_on_side=temp,curr_power_on_
speed=fast,pend_power_on_speed=fast,curr_power_on_speed_override=none,pend_power_
on_speed_override=none,power_on_type=power on,power_on_option=standby,power_on_
lpar_start_policy=userinit,pend_power_on_option=standby,pend_power_on_lpar_start_
policy=userinit,power_on_method=02,power_on_attr=0000,sp_boot_attr=0000,sp_boot_
major_type=08,sp_boot_minor_type=01,sp_version=00030030,mfg_default_config=0,curr_
mfg_default_ipl_source=a,pend_mfg_default_ipl_source=a,curr_mfg_default_boot_
mode=norm,pend_mfg_default_boot_mode=norm
```

Mapping

For each detected IBM Pseries Frame, a Hypervisor CI is created with the set name attribute IBM Hypervisor.

The output of this command is used to fill in the attributes of the **IBM PSeries Frame** CI:

CMD Output Attribute	CI Name	CI Attribute
Name	IBM PSeries Frame	Name
serial_number	IBM PSeries Frame	Host Key
cod_proc_capable	IBM PSeries Frame	CPU Capacity on Demand Capable
cod_mem_capable	IBM PSeries Frame	Memory Capacity on Demand Capable
huge_page_mem_capable	IBM PSeries Frame	Huge Memory Page Capable
max_lpars	IBM PSeries Frame	Max LPARs

CMD Output Attribute	CI Name	CI Attribute
Status	IBM PSeries Frame	Frame State
micro_lpar_capable	IBM PSeries Frame	Micro LPAR Capable
service_lpar_id	IBM PSeries Frame	Service LPAR ID
service_lpar_name	IBM PSeries Frame	Service LPAR Name

lshwres -r proc --level sys -m '<Managed System Name>'

Output

```
configurable_sys_proc_units=4.0,curr_avail_sys_proc_units=1.4, pend_avail_sys_proc_
units=1.4,installed_sys_proc_units=4.0, max_capacity_sys_proc_
units=deprecated,deconfig_sys_proc_units=0, min_proc_units_per_virtual_
proc=0.1,max_virtual_procs_per_lpar=64,max_procs_per_lpar=4,max_curr_virtual_procs_
per_aixlinux_lpar=64,max_curr_virtual_procs_per_vios_lpar=64, max_curr_virtual_
procs_per_os400_lpar=64,max_curr_procs_per_aixlinux_lpar=4, max_curr_procs_per_
vios_lpar=4,max_curr_procs_per_os400_lpar=4, max_shared_proc_pools=1
```

Mapping

The output of this command is used to fill in the attributes of the **IBM PSeries Frame** CI:

CMD Output Attribute	CI Name	CI Attribute
min_proc_units_per_virtual_proc	IBM PSeries Frame	Min CPU Units per Virtual CPU
curr_avail_sys_proc_units	IBM PSeries Frame	Current Available CPU Units
max_shared_proc_pools	IBM PSeries Frame	Max Shared CPU Pools
configurable_sys_proc_units	IBM PSeries Frame	Configurable CPU Units
installed_sys_proc_units	IBM PSeries Frame	Installed CPU Units
pend_avail_sys_proc_units	IBM PSeries Frame	Pending Available CPU Units
max_procs_per_lpar	IBM PSeries Frame	Max CPUs per LPAR
max_virtual_procs_per_lpar	IBM PSeries Frame	Max Virtual CPUs per LPAR

lshwres -r mem --level sys -m '<Managed System Name>'

Output

```
configurable_sys_mem=32768,curr_avail_sys_mem=1344,pend_avail_sys_mem=1344,
installed_sys_mem=32768,max_capacity_sys_mem=deprecated,deconfig_sys_mem=0, sys_
firmware_mem=704,mem_region_size=64,configurable_num_sys_huge_pages=0, curr_avail_
num_sys_huge_pages=0,pend_avail_num_sys_huge_pages=0, max_num_sys_huge_
pages=1,requested_num_sys_huge_pages=0,huge_page_size=16384, max_mem_pools=0
```

Mapping

The output of this command is used to fill in the attributes of the **IBM PSeries Frame** CI:

CMD Output Attribute	CI Name	CI Attribute
configurable_sys_mem	IBM PSeries Frame	Configurable System Memory
max_num_sys_huge_pages	IBM PSeries Frame	Max Number of Huge Pages
huge_page_size	IBM PSeries Frame	Huge Page Size
sys_firmware_mem	IBM PSeries Frame	Firmware Memory
mem_region_size	IBM PSeries Frame	Memory Region Size
curr_avail_sys_mem	IBM PSeries Frame	Current Available Memory
installed_sys_mem	IBM PSeries Frame	Installed Memory
requested_num_sys_huge_pages	IBM PSeries Frame	Requested Number of Huge Pages
pend_avail_sys_mem	IBM PSeries Frame	Pending Available Memory

lshwres -r proc --level pool -m '<Managed System Name>'

Output

```
configurable_pool_proc_units=4.0,curr_avail_pool_proc_units=1.4,pend_avail_pool_
proc_units=1.4
```

Mapping

If there are no user-defined pools, the **pool_id** parameter does not appear in the output (**pool_id** is considered by the system to be zero by default).

The output of this command is used to fill in the attributes of the **IBM Processor Pool** CI:

CMD Output Attribute	CI Name	CI Attribute
curr_avail_pool_proc_units	IBM Processor Pool	CPU Pool Available Physical CPUs

CMD Output Attribute	CI Name	CI Attribute
configurable_pool_proc_units	IBM Processor Pool	CPU Pool Configurable Physical CPUs
pend_avail_pool_proc_units	IBM Processor Pool	CPU Pool Pending Available Physical CPUs
pool_id	IBM Processor Pool	Name

lssyscfg -r lpar -m '<Managed System Name>'

Output

```
name=somelparname1,lpar_id=5,lpar_env=aixlinux,state=Running,resource_config=1,os_
version=Unknown,logical_serial_num=65B922G5,default_
profile=somedefaultprofilename1,curr_profile=somelparprofilename1,work_group_
id=none,shared_proc_pool_util_auth=1,allow_perf_collection=1,power_ctrl_lpar_
ids=none,boot_mode=sms,lpar_keylock=norm,auto_start=0,redundant_err_path_
reporting=0
```

Mapping

The output of this command is used to fill in the attributes of the **IBM LPAR Profile** CI:

CMD Output Attribute	CI Name	CI Attribute
logical_serial_num	IBM LPAR Profile	LPAR Serial Number
boot_mode	IBM LPAR Profile	LPAR Profile Boot Mode
auto_start	IBM LPAR Profile	LPAR Profile Auto Start
work_group_id	IBM LPAR Profile	LPAR Profile Workgroup ID
default_profile	IBM LPAR Profile	LPAR default profile name
curr_profile	IBM LPAR Profile	LPAR profile name
power_ctrl_lpar_ids	IBM LPAR Profile	LPAR power control ids
State	IBM LPAR Profile	Lpar state
lpar_env	IBM LPAR Profile	Lpar type
lpar_id	IBM LPAR Profile	LPAR ID
Name	IBM LPAR Profile	LPAR Name

lssyscfg -r prof -m '<Managed System Name>'

Output

```
name=name1,lpar_name=name2,lpar_id=5,lpar_env=aixlinux,
all_resources=0,min_mem=4096,desired_mem=8192,max_mem=8192,
min_num_huge_pages=0,desired_num_huge_pages=0,
max_num_huge_pages=0,proc_mode=shared,min_proc_units=0.3,
desired_proc_units=0.5,max_proc_units=1.0,min_procs=1,
desired_procs=2,max_procs=2,sharing_mode=uncap,
uncap_weight=128,io_slots=none,lpar_io_pool_ids=none,
max_virtual_slots=10,"virtual_serial_adapters=0/server/1/
any//any/1,1/server/1/any//any/1","virtual_scsi_adapters=5/
client/1/l11s12vio1/13/1,6/client/1/l11s12vio1/14/1,7/client
/1/l11s12vio1/15/1",virtual_eth_adapters=2/0/1//0/1,
hca_adapters=none,boot_mode=norm,conn_monitoring=1,auto_start=0,
power_ctrl_lpar_ids=none,work_group_id=none,redundant_err_path_reporting=0
name=name3,lpar_name=name4,lpar_id=4,lpar_env=aixlinux,all_resources=0,
min_mem=4096,desired_mem=10240,max_mem=10240,min_num_huge_pages=0,
desired_num_huge_pages=0,max_num_huge_pages=0,proc_mode=shared,
min_proc_units=0.3,desired_proc_units=0.7,max_proc_units=1.0,
min_procs=1,desired_procs=2,max_procs=2,sharing_mode=uncap,
uncap_weight=128,io_slots=none,lpar_io_pool_ids=none,
max_virtual_slots=10,"virtual_serial_adapters=0/server
/1/any//any/1,1/server/1/any//any/1",
"virtual_scsi_adapters=5/client/1/l11s12vio1/10/1,6/
client/1/l11s12vio1/11/1,7/client/1/l11s12vio1/12/1",
virtual_eth_adapters=2/0/2//0/1,hca_adapters=none,boot_mode=norm,
conn_monitoring=1,auto_start=0,power_ctrl_lpar_ids=none,
work_group_id=none,redundant_err_path_reporting=0
```

Mapping

The output of this command is used to fill in the attributes of the **IBM LPAR Profile** CI:

CMD Output Attribute	CI Name	CI Attribute
sharing_mode	IBM LPAR Profile	LPAR Profile Sharing Mode
proc_mode	IBM LPAR Profile	LPAR Profile CPU Mode
uncap_weight	IBM LPAR Profile	LPAR Profile Uncapped Weight

CMD Output Attribute	CI Name	CI Attribute
desired_num_huge_pages	IBM LPAR Profile	LPAR Profile Desired Number of Huge Memory Pages
min_num_huge_pages	IBM LPAR Profile	LPAR Profile Minimum Number of Huge Memory Pages
max_procs	IBM LPAR Profile	LPAR Profile Maximum Number of CPUs
desired_procs	IBM LPAR Profile	LPAR Profile Desired Number of CPUs
min_proc_units	IBM LPAR Profile	LPAR Profile Minimum Physical CPUs
max_mem	IBM LPAR Profile	LPAR Profile Maximum memory
conn_monitoring	IBM LPAR Profile	LPAR Profile Connection Monitoring Enabled
min_mem	IBM LPAR Profile	LPAR Profile Minimum Memory on this LPAR
max_virtual_slots	IBM LPAR Profile	LPAR Profile Maximum Number of Virtual Slots
redundant_err_path_ reporting	IBM LPAR Profile	LPAR Profile Redundant Error Path Reporting
max_num_huge_pages	IBM LPAR Profile	LPAR Profile Maximum Number of Huge Memory Pages
min_procs	IBM LPAR Profile	LPAR Profile Minimum Number of CPUs
max_proc_units	IBM LPAR Profile	LPAR Profile Maximum Physical CPUs
io_slots	IBM LPAR Profile	LPAR Profile IO Slots
lpar_io_pool_ids	IBM LPAR Profile	LPAR Profile IO Pool IDs

CMD Output Attribute	CI Name	CI Attribute
desired_proc_units	IBM LPAR Profile	LPAR Profile Desired Physical CPUs
desired_mem	IBM LPAR Profile	LPAR Profile Memory Requested by this LPAR
virtual_serial_adapters	IBM LPAR Profile	LPAR Profile Virtual Serial Adapters

lshwres -r virtualio --rsubtype eth --level lpar -m '<Managed System Name>'

Output

```
lpar_name=name1,lpar_id=1,slot_num=2,state=1,is_required=1,is_trunk=1,trunk_
priority=1, ieee_virtual_eth=0,port_vlan_id=1,addl_vlan_ids=,mac_addr=765920001002
lpar_name=l11s12vio1,lpar_id=1,slot_num=3,state=1,is_required=1,is_trunk=1,trunk_
priority=1, ieee_virtual_eth=0,port_vlan_id=2,addl_vlan_ids=,mac_addr=765920001003
lpar_name=name2,lpar_id=2,slot_num=2,state=1,is_required=1,is_trunk=0,ieee_virtual_
eth=0, port_vlan_id=1,addl_vlan_ids=,mac_addr=765920002002
lpar_name=name3,lpar_id=3,slot_num=2,state=1,is_required=1,is_trunk=0,ieee_virtual_
eth=0, port_vlan_id=1,addl_vlan_ids=,mac_addr=765920003002
lpar_name=name4,lpar_id=4,slot_num=2,state=1,is_required=1,is_trunk=0,ieee_virtual_
eth=0, port_vlan_id=2,addl_vlan_ids=,mac_addr=765920004002
lpar_name=name5,lpar_id=5,slot_num=2,state=1,is_required=1,is_trunk=0,ieee_virtual_
eth=0, port_vlan_id=1,addl_vlan_ids=,mac_addr=765920005002
```

Mapping

The mac_addr attribute is represented in the Dec form without leading zeros. This value is transformed to the Hex value and left padded with missing zeros, to assure a proper representation of the MAC address in the CMDB.

Based on the MAC address, the virtual NICs are created and attached to the corresponding LPAR or VIO server, and are described by **Lpar_name** or **Lpar_id**. The **Vlan** CI is created based on **vlan_id** or **addl_vlan_ids** and is linked to the ports of the interfaces. The root container for the VLAN is a specific IBM PSeries Frame (Managed System).

CMD Output Attribute	CI Name	CI Attribute
port_vlan_id/addl_vlan_ids	VLAN	Vlan Number
IBM PSeries Frame CMDB ID	VLAN	Root Container
mac_addr (converted to Hex if needed and normalized)	Interface	MAC Address

lshwres -r virtualio --rsubtype scsi -m '<Managed System Name>'

Output

```
lpar_name=vioname1,lpar_id=1,slot_num=15,state=1,is_required=0,adapter_
type=server,remote_lpar_id=5,remote_lpar_name=lparname1,remote_slot_num=7
lpar_name=vioname1,lpar_id=1,slot_num=14,state=1,is_required=0,adapter_
type=server,remote_lpar_id=5,remote_lpar_name=lparname2,remote_slot_num=6
lpar_name=vioname1,lpar_id=1,slot_num=13,state=1,is_required=0,adapter_
type=server,remote_lpar_id=5,remote_lpar_name=lparname2,remote_slot_num=5
```

Mapping

The lpar_name and lpar_id attributes are always the name and ID of the VIO server that creates and grants the Virtual SCSI to the LPARs. The SCSI Adapter on the LPAR is identified by its slot number and the LPAR name it belongs to.

CMD Output Attribute	CI Name	CI Attribute
Slot_num/remote_slot_num	SCSI	Slot Number
Host ID with name <lpar_name> or <Remote LPAR Name>	SCSI	Root Container

lshwres -r proc --level lpar -m '<Managed System Name>'

Output

```
lpar_name=name1,lpar_id=5,curr_shared_proc_pool_id=0,curr_proc_mode=shared,curr_
min_proc_units=0.3,curr_proc_units=0.5,curr_max_proc_units=1.0,curr_min_
procs=1,curr_procs=2,curr_max_procs=2,curr_sharing_mode=uncap,curr_uncap_
weight=128,pend_shared_proc_pool_id=0,pend_proc_mode=shared,pend_min_proc_
units=0.3,pend_proc_units=0.5,pend_max_proc_units=1.0,pend_min_procs=1,pend_
procs=2,pend_max_procs=2,pend_sharing_mode=uncap,pend_uncap_weight=128,run_proc_
units=0.5,run_procs=2,run_uncap_weight=128
```

Mapping

Using the "lpar_name"/"lpar_id" along with the "curr_shared_proc_pool_id" from the output we can create corresponding links to the particular Shared Processor Pool ("IBM Processor Pool") the LPar uses. In case of the dedicated ("ded") CPU we will create links to the spare processors.

lshwres -r io --rsubtype slot -m '<Managed System Name>'

Output

```
unit_phys_loc=XXXXX.XXX.XXXXXXX,bus_id=2,phys_loc=C3,drc_index=21010002,lpar_
name=name1,lpar_id=1,slot_io_pool_id=none,description=RAID Controller,feature_
```

```
codes=none,pci_vendor_id=1069,pci_device_id=B166,pci_subs_vendor_id=1014,pci_subs_
device_id=0278,pci_class=0104,pci_revision_id=04,bus_grouping=0,iop=0,parent_slot_
drc_index=none,drc_name=XXXXX.XXX.XXXXXXX-XX-XX
```

Mapping

The output of this command is used to create the **I/O Slot** CI. Using the name and ID of the LPAR, discovery creates the relationship to the particular LPAR that is using the slot.

CMD Output Attribute	CI Name	CI Attribute
Description	I/O Slot	Name of the Slot
bus_id	I/O Slot	Slot Bus ID
phys_loc	I/O Slot	Slot Physical Location on Bus
pci_revision_id	I/O Slot	Slot PCI Revision ID
bus_grouping	I/O Slot	Slot Bus Grouping
pci_device_id	I/O Slot	Slot PCI Device ID
unit_phys_loc	I/O Slot	Slot Physical Location
parent_slot_drc_index	I/O Slot	Slot Parent Slot DRC Index
drc_index	I/O Slot	Slot DRC Index
pci_subs_vendor_id	I/O Slot	Slot PCI Subslot Vendor ID
pci_class	I/O Slot	Slot PCI Class
slot_io_pool_id	I/O Slot	Slot IO Pool ID
pci_vendor_id	I/O Slot	Slot PCI Vendor ID
drc_name	I/O Slot	Slot DRC Name
feature_codes	I/O Slot	Slot Feature Codes
pci_subs_device_id	I/O Slot	Slot PCI Subslot Device ID

VIO Server Side Commands

This section includes the following commands:

/usr/ios/cli/ioscli lsdev -dev 'ent*' -field name physloc -fmt

Output

```
ent0: U100C.001.DQDE777-P1-C4-T1
ent1:U100C.001.DQDE777-P1-C4-T2
ent2:U100C.001.DQDE777-P1-C4-T3
ent16:
```

```
ent17:
ent18:
ent19:
ent20:
```

Mapping

The interface names and physical location of the particular interface are the output of this command. The output is split at the colon character (**:**) line by line; the first part is the interface name and the last is the physical location. A physical location is not always present, for example, it is not set for the SEA and Link Aggregation Interface. The physical location value is used to create a link from the physical NIC to the I/O slot.

ioscli entstat -all '<Interface Name>' | grep -E "ETHERNET STATISTICS|Device Type|Hardware Address

Example: ioscli entstat -all 'ent16'| grep -E "ETHERNET STATISTICS|Device Type|Hardware Address

Output

```
ETHERNET STATISTICS (ent16) :
Device Type: Shared Ethernet Adapter
Hardware Address: 00:1B:64:91:74:55
ETHERNET STATISTICS (ent14) :
Device Type: EtherChannel
Hardware Address: 00:1B:64:91:74:55
ETHERNET STATISTICS (ent0) :
Device Type: 2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)
Hardware Address: 00:1a:64:91:74:44
ETHERNET STATISTICS (ent2) :
Device Type: 2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)
Hardware Address: 00:1B:64:91:74:55
ETHERNET STATISTICS (ent4) :
Device Type: Virtual I/O Ethernet Adapter (l-lan)
Hardware Address: 46:61:fa:d4:bf:0b
```

Mapping

UCMDB Version 8.0x: There cannot be two interfaces with the same MAC on a single machine. In this case the MAC Address attribute for the first interface only takes the value of the MAC address, while the other interfaces contain an underscore (_) and interface index. For example, for the above output interface **ent0** is reported with MAC Address set to **00:1B:64:91:74:55** while interface **ent2** is reported with MAC Address set to **00:1B:64:91:74:55_2**.

UCMDB Version 9.0x: This limitation is not relevant so the topology is reported as is.

CMD Output Attribute	CI Name	CI Attribute
ETHERNET STATISTICS line	Interface	Name
Hardware Address	Interface	Mac Address
Device Type	Interface	Description
ETHERNET STATISTICS line when Device Type value is EtherChannel	Interface Aggregation	Name
ETHERNET STATISTICS line when Device Type value is Shared Ethernet Adapter	IBM SEA	Name

ioscli lsdev -dev '<Interface Name>' -attr

Example: ioscli lsdev -dev 'ent16' -attr

Output

```
attribute value description user_settable
adapter_names ent0,ent4 EtherChannel Adapters True
alt_addr 0x000000000000 Alternate EtherChannel Address True
auto_recovery yes Enable automatic recovery after failover True
backup_adapter NONE Adapter used when whole channel fails True
hash_mode default Determines how outgoing adapter is chosen True
mode standard EtherChannel mode of operation True
netaddr 0 Address to ping True
noloss_failover yes Enable lossless failover after ping failure True
num_retries 3 Times to retry ping before failing True retry_time 1 Wait time (in
seconds) between pings True
use_alt_addr no Enable Alternate EtherChannel Address True
use_jumbo_frame no Enable Gigabit Ethernet Jumbo Frames True
```

Mapping

The adapter_names attribute value is used to create links to the back-up devices.

The value of Media Speed represents both Duplex and the connection Speed.

CMD Output Attribute	CI Name	CI Attribute
media_speed	Interface Index	Speed

ioscli lsmap -all -net

Output

```
SVEA Physloc
------ --------------------------------------------
ent4 U1000.E4A.06FB0D1-V1-C11-T1
SEA ent16
Backing device ent14
Status Available
Physloc
SVEA Physloc
------ --------------------------------------------
ent9 U1000.E4A.06FB0D1-V1-C16-T1
SEA ent21
Backing device ent12
Status Available
Physloc U1000.001.DQD3693-P1-C7-T3
```

Mapping

This command is used to determine the relation between the interfaces and to identify their types.

CMD Output Attribute	CI Name	CI Attribute
SEA	SEA Adapter	Name
Backing Device	Link Aggregation / Interface	Name
SVEA	Interface (virtual)	Name

ioscli lsdev -dev fcs* -field name physloc description -fmt

Output

```
fcs0:U1000.001.DQDE996-P1-C1-T1:4Gb FC PCI Express Adapter (df1000fe)
fcs1:U1000.001.DQDE996-P1-C1-T2:4Gb FC PCI Express Adapter (df1000fe)
fcs2:U1000.001.DQDE996-P1-C2-T1:4Gb FC PCI Express Adapter (df1000fe)
fcs3:U1000.001.DQDE996-P1-C2-T2:4Gb FC PCI Express Adapter (df1000fe)
```

Mapping

The output of this command represents the Fibre Channel Host Adapters on the VIO server. This output retrieves the FC Name and FC Physical Path, which are used to create a link to the I/O slot on the PFrame, and an FC Interface Description.

CMD Output Attribute	CI Name	CI Attribute
First token	Fibre Channel HBA	Name
Third token	Fibre Channel HBA	Description

ioscli lsdev | grep proc

Output

```
proc0   Available   Processor
proc2   Available   Processor
proc4   Available   Processor
proc6   Available   Processor
```

Mapping

The output of this command shows discovered CPU indices. In this case 0, 2, 4, and 6.

ioscli lsdev -dev sysplanar0 -vpd | grep PROC

Output

```
2-WAY PROC CUOD :
2-WAY PROC CUOD :
2-WAY PROC CUOD :
2-WAY PROC CUOD :
```

Mapping

The output of this command shows the number of cores discovered. The "2" in the output indicates 2 CPU cores.

ioscli lsdev -dev proc<index> -attr

Output

```
attribute   value          description           user_settable
frequency   1654344000     Processor Speed       False
smt_enabled true           Processor SMT enabled False
smt_threads 2              Processor SMT threads False
state       enable         Processor state       False
type        PowerPC_POWER5 Processor type        False
```

Mapping

CMD Output Attribute	CI Name	CI Attribute
frequency	CPU	speed
type	CPU	model

lspv

Output

```
NAME PVID VG STATUS
hdisk0 001fb2d15d794e0d rootvg active
hdisk1 001fb2d18f1f7f0c clientvg active
```

Mapping

This command retrieves the relation between the Physical Volume and the Volume Group, then a link is created from the Volume Group to the Physical Volume.

CMD Output Attribute	CI Name	CI Attribute
VG	Physical Volume	Name
VG	Fibre Channel HBA	Name

lsvg

Output

```
rootvg clientvg
```

Mapping

This command retrieves the list of all volume groups that are present on the VIO server.

lsvg <Volume Group Name>

Output

```
VOLUME GROUP: rootvg
VG IDENTIFIER: 001fb2d10005d9000000011a5d795185
VG STATE: active
PP SIZE: 256 megabyte(s)
VG PERMISSION: read/write
TOTAL PPs: 520 (133120 megabytes)
MAX LVs: 256
FREE PPs: 372 (95232 megabytes)
LVs: 13
USED PPs: 148 (37888 megabytes)
OPEN LVs: 11
QUORUM: 2 (Enabled)
TOTAL PVs: 1
VG DESCRIPTORS: 2
STALE PVs: 0
STALE PPs: 0
ACTIVE PVs: 1
AUTO ON: yes
MAX PPs per VG: 32512
MAX PPs per PV: 1016
MAX PVs: 32
LTG size (Dynamic): 256 kilobyte(s)
AUTO SYNC: no
HOT SPARE: no
BB POLICY: relocatable
```

Mapping

This command retrieves the values for the Volume Group CI attributes.

CMD Output Attribute	CI Name	CI Attribute
VOLUME GROUP	Volume Group	Name
STATE	Volume Group	Volume Group State
VG IDENTIFIER	Volume Group	Volume Group ID

lsvg -lv <Volume Group Name>

Output

```
rootvg:
LV NAME TYPE LPs PPs PVs LV STATE MOUNT POINT
hd5 boot 1 1 1 closed/syncd N/A
hd6 paging 2 2 1 open/syncd N/A
paging00 paging 4 4 1 open/syncd N/A
hd8 jfs2log 1 1 1 open/syncd N/A
hd4 jfs2 1 1 1 open/syncd /
hd2 jfs2 10 10 1 open/syncd /usr
hd9var jfs2 3 3 1 open/syncd /var
hd3 jfs2 10 10 1 open/syncd /tmp
hd1 jfs2 40 40 1 open/syncd /home
hd10opt jfs2 4 4 1 open/syncd /opt
lg_dumplv sysdump 4 4 1 open/syncd N/A
VMLib_LV jfs2 56 56 1 open/syncd /var/vio/VMLib
Ilv jfs2 12 12 1 closed/syncd /export/lbm
```

Mapping

This command retrieves the list of all Logical Volumes that are part of the particular Volume Group, as well as the mount points if any exist. This information enables the creation of a link from the Volume Group to the Logical Volume.

CMD Output Attribute	CI Name	CI Attribute
LV Name	Logical Volume	Name
Mount Point	Disk (FS)	Name
Type	Disk	Type

lsvg -pv <Logical Volume Group>

Output

```
rootvg:
PV_NAME PV STATE TOTAL PPs FREE PPs FREE DISTRIBUTION
hdisk0 active 520 372 103..30..31..104..104
```

Mapping

This command retrieves the list of the Physical Volumes in the Volume Group. This information enables the creation of a link between the Physical Volume and the Volume Group.

lslv <Logical Volume Name>

Output

```
LOGICAL VOLUME: lv1
VOLUME GROUP: clientvg
LV IDENTIFIER: 000fb1d10230d9000000011b8f1f8187.1
PERMISSION: read/write
VG STATE: active/complete
LV STATE: opened/syncd
TYPE: jfs
WRITE VERIFY: off
MAX LPs: 32512
PP SIZE: 512 megabyte(s)
COPIES: 1
SCHED POLICY: parallel
LPs: 70
PPs: 70
STALE PPs: 0
BB POLICY: non-relocatable
INTER-POLICY: minimum
RELOCATABLE: yes
INTRA-POLICY: middle
UPPER BOUND: 1024
MOUNT POINT: N/A
LABEL: None
MIRROR WRITE
```

```
CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?: NO
DEVICESUBTYPE : DS_LVZ
```

Mapping

This command retrieves information about the Logical Volume parameters, which are mapped to the attributes of the Logical Volume CI.

CMD Output Attribute	CI Name	CI Attribute
LOGICAL VOLUME	Logical Volume	Name
LV IDENTIFIER	Logical Volume	Logical Volume ID
LV STATE	Logical Volume	Logical Volume Status
Type	Logical Volume	Logical Volume File System Type

ioscli lsmap -all

Output

```
SVSA Physloc Client Partition ID
--------------- -------------------------------------------- ------------------
vhost0 U1000.E4A.06FB0D1-V1-C21 0x00000002
VTD vtopt0
Status Available
LUN 0x8100000000000000
Backing device /var/vio/VMLib/bootcd_rh5
Physloc
SVSA Physloc Client Partition ID
--------------- -------------------------------------------- ------------------
vhost3 U1000.E4A.06FB0D1-V1-C31 0x00000002
VTD vtscsi0
Status Available
LUN 0x8100000000000000
Backing device os_ lv1
Physloc
VTD vtscsi1
Status Available
LUN 0x8200000000000000
Backing device p01_lv1
Physloc
VTD vtscsi8
Status Available
LUN 0x8300000000000000
```

```
Backing device p01_lv2
Physloc
```

Mapping

This command retrieves the relation from the vSCSI to the exact backing device, which is usually a Volume or a Volume Group.

CMD Output Attribute	CI Name	CI Attribute
SVSA	SCSI	Name
C<Number>	SCSI	Slot Number
Backing Device	LV/PV/FS	Name

LPAR Side Commands

This section includes the following command:

lscfg

Output

```
INSTALLED RESOURCE LISTThe following resources are
installed on the machine.+/- = Added or deleted from
Resource List.* = Diagnostic support not available.
Model Architecture: chrp
Model Implementation: Multiple Processor, PCI bus + sys0
System Object+ sysplanar0 System Planar* vio0
Virtual I/O Bus* vsa0 U1000.505.062136A-V1-C0
LPAR Virtual Serial Adapter* vty0 U1000.505.062136A-V1-C0-L0
Asynchronous Terminal* pci2 U1000.001.AAA0757-P1
PCI Bus* pci1 U1000.001.AAA0757-P1
PCI Bus* pci0 U1000.001.AAA0757-P1
PCI Bus* pci3 U1000.001.AAA0757-P1
PCI Bus+ ent0 U1000.001.AAA0757-P1-T1
2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)+ ent1
U1000.001.AAA0757-P1-T2
2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)* pci4
U1000.001.AAA0757-P1
PCI Bus+ usbhc0 U1000.001.AAA0757-P1
USB Host Controller (33103500)+ usbhc1 U1000.001.AAA0757-P1
USB Host Controller (33103500)* pci5   U1000.001.AAA0757-P1
PCI Bus* ide0  U1000.001.AAA0757-P1-T10
ATA/IDE Controller Device+ cd0 U1000.001.AAA0757-P1-D3
IDE DVD-ROM Drive* pci6  U1000.001.AAA0757-P1
PCI Bus+ sisscsia0 U1000.001.AAA0757-P1
PCI-X Dual Channel Ultra320
SCSI Adapter+ scsi0   U1000.001.AAA0757-P1-T5
PCI-X Dual Channel Ultra320
SCSI Adapter bus+ scsi1 U1000.001.AAA0757-P1-T9
PCI-X Dual Channel Ultra320
SCSI Adapter bus+ hdisk0 U1000.001.AAA0757-P1-T9-L5-L0 16 Bit LVD
SCSI Disk Drive (146800 MB)+ hdisk1 U1000.001.AAA0757-P1-T9-L8-L0
16 Bit LVD
SCSI Disk Drive (146800 MB)+
ses0  U1000.001.AAA0757-P1-T9-L15-L0
SCSI Enclosure Services Device+
L2cache0 L2 Cache+ mem0 Memory+ proc0 Processor
```

Created/Changed Entities

Entity Name	Entity Type	Entity Description
IBM HMC	CI Type	HMC software
IBM LPar Profile	CI Type	LPar configuration
IBM Processor Pool	CI Type	Shared Processor Pool
IBM PSeries Frame	CI Type	PSeries Frame/Managed System
Interface Aggregation	CI Type	Link Aggregation
I/O Slot	CI Type	I/O Slot on the Frame
SEA Adapter	CI Type	Virtual Eth interface on a VIO Server
IBM Processor Pool > containment > CPU	Valid Link	
I/O Slot > containment > Fibre Channel HBA	Valid Link	
I/O Slot> containment > Network Interface	Valid Link	
I/O Slot > containment > SCSI Adapter	Valid Link	
IBM HMC > manage > IBM PSeries Frame	Valid Link	
Interface Aggregation > membership > Network Interface	Valid Link	
Network Interface > realization > Network Interface	Valid Link	
Network Interface > usage > SEA Adapter	Valid Link	
SEA Adapter > usage > Network Interface	Valid Link	
IBM Virtualization by Shell	Job	Performs Virtualization based discovery

Entity Name	Entity Type	Entity Description
IBM LPAR and VIO Server Topology by Shell	Job	Performs LPAR and VIO Server side discovery
IBM_VIRTUALIZATION_BY_SHELL_ PATTERN	Adapter	Adapter for the IBM Virtualization by Shell job
IBM_LPAR_VIO_BY_SHELL	Adapter	Adapter for the IBM LPAR and VIO Server Topology by Shell job
ibm_hmc_by_shell	Script	General HMC side discovery script
ibm_hmc_lib	Script	Common Data Objects and Procedures for both new Jobs
ibm_lpar_or_vio_by_shell	Script	General VIO Server and LPAR discovery script
ibm_hmc_by_shell.xml	query	Trigger query for the IBM Virtualization by Shell job
ibm_lpar_or_vio_trigger_tql.xml	query	Trigger query for the IBM LPAR and VIO Server Topology by Shell job
IBM HMC Topology.xml	query	Query (TQL) for the IBM HMC Topology view
IBM Storage Topology.xml	query	Query (TQL) for the IBM Storage Topology view
IBM HMC Topology.xml	View	
IBM Storage Topology.xml	View	
lpar_boot_mode	Type	Supported boot modes
lpar_cpu_mode	Type	CPU Sharing modes
lpar_sharing_mode	Type	LPAR cap/uncap sharing modes
lpar_state	Type	Possible LPAR states
lpar_type	Type	Possible LPAR types

Troubleshooting and Limitations

This section describes troubleshooting and limitations for IBM Virtual Discovery.

- It is possible to configure the Partition Migration of an LPAR to the PFrame. This is supported only in Power Series v6, and is presently not supported by IBM Virtual Discovery.
- VIO Server on Linux OS is not supported.

Chapter 12: Oracle VM Server for SPARC Technology Discovery

Overview

The Oracle VM Server for SPARC Technology Discovery allows the discovery of Oracle LDOM (Logical Domains) or Oracle VM Server for SPARC technology.

Supported Versions

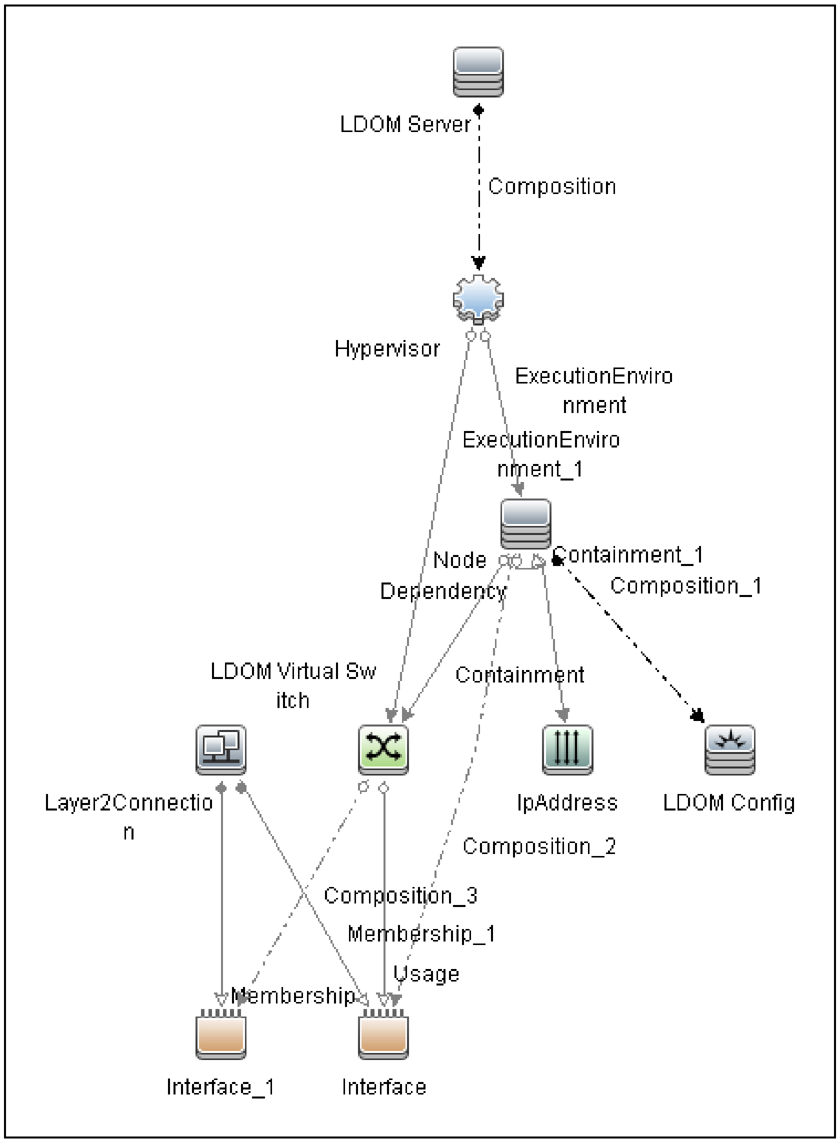
Oracle VM Server for SPARC Technology Discovery supports LDOM versions 1.0-1.3 and 2.0, and Oracle VM Server for SPARC versions 2.0-2.1.

Topology

This section displays the following topology maps:

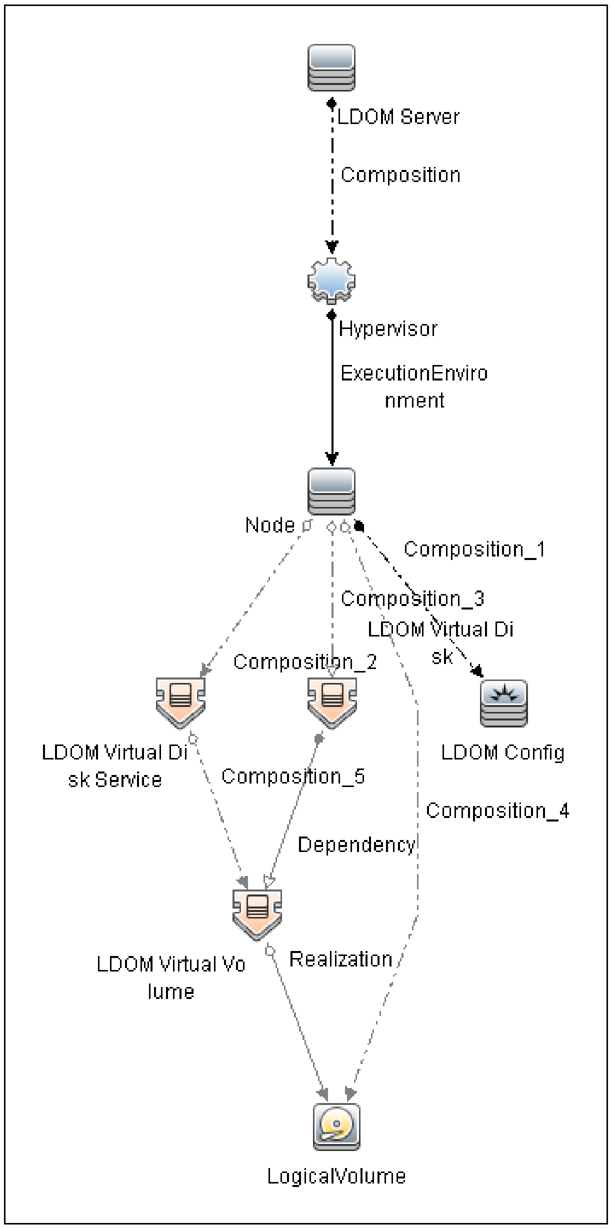
- "LDOM Networking and General Topology" below
- "LDOM Storage Topology" on the next page

LDOM Networking and General Topology



Note: For a list of discovered CITs, see "Discovered CITs" on page 184.

LDOM Storage Topology



Note: For a list of discovered CITs, see"Discovered CITs" on page 184.

How to Discover Oracle VM Server for SPARC Technology

1. **Prerequisites - General**

   a. Shell connectivity to the control domain.

   b. If required, configure **sudo** on each target host to allow execution of the following commands.

   ```
   /opt/SUNWldm/bin/ldm list*
   /usr/sbin/ldm list*
   ```

   The path is dependent on where the ldm command is located.

2. **Prerequisites - Setup protocol credentials**

   Setup one of the following protocols:

   - SSH
   - Telnet

     For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

3. **Run the discovery**

   - Run the **Range IPs by ICMP** job to discover the target IPs.
   - Run the **Host Connection by Shell** job to discover the target host and shell connectivity to it.
   - Run **Host Applications by Shell** job to discover applications of the target host, including the **Logical Domains Manager** application.
   - Run **Oracle VM Server for SPARC Technology by Shell** job in order to discover the topology of the target LDOM server.

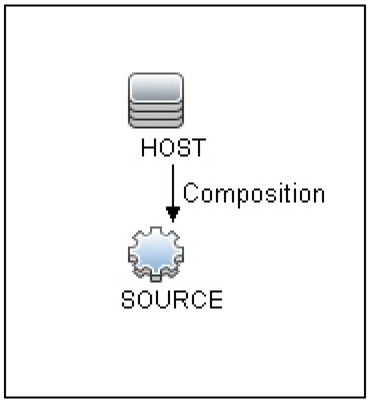
Oracle_VM_Server_for_SPARC_Technology_by_Shell Adapter

This section includes the following information:

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
hostId	${SOURCE.root_container}
ip_address	${SOURCE.application_ip}
protocol	${SOURCE.root_class}

Used Scripts

- ldom.py
- ldom_by_shell.py
- ldom_discover.py
- ldom_report.py
- networking.py
- solaris_networking.py

Discovered CITs

- Composition
- Containment
- Dependency
- ExecutionEnvironment
- Hypervisor
- Interface

- IpAddress
- Layer2Connection
- LDOM Resource
- Logical Volume
- Membership
- Node
- Realization

Parameters

Name	Description
match_domain_names_to_hostnames	When enabled, the discovery reports guest LDOMs, with their hostnames set to domain names, which may aid in the reconciliation of hosts. **Default**: false.

Oracle VM Server for SPARC Technology by Shell Job

This section includes the following information:

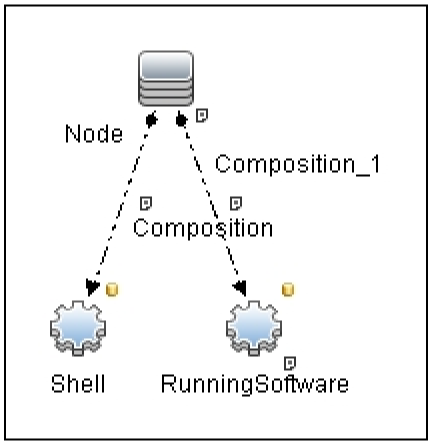
- "Adapter" below
- "Trigger Query" on the next page

Adapter

This job uses the **Oracle_VM_Server_for_SPARC_Technology_by_shell** adapter.

Trigger Query

Name: ldom_control_domain_by_shell



Node Name	Condition
Node	None
Shell	CI Type Equal "SSH" or CI Type Equal "Telnet"
RunningSoftware	DiscoveredProductName Equal "Logical Domains Manager"

Discovery Flow

This section describes the discovery flow of the Oracle VM Server for SPARC Technology by Shell job.

General

- Discovery is performed by using the shell of the control domain
- The single command **ldm** of the control domain provides most of the required configuration information
- Guest domains:
  - Are completely isolated.
  - May have no network connectivity to control domain.
  - Can have an OS different from Solaris.

    Note: For versions of LDOM below 2.0, and for guest OS different from Solaris, it is not possible to know whether it is a guest domain or a regular host.

    Accordingly, no specific discovery by guest domains is performed.
- Only domains that are in active or bound states are discovered, since for domains in other states the configuration may be incomplete or stale.

Oracle VM Server for SPARC Technology by Shell Job Flow

- **Get version of Logical Domains Manager**

  The **ldm** command is executed to get the version of **Logical Domains Manager**. See "Obtaining version information of Logical Domains manager" on the next page. To run **ldm**:

  - Make sure the **ldm** command is present, otherwise it is not a control domain and further discovery is impossible.
  - Get the proper path to the **ldm** command, which can be located under **/opt/SUNWldm/bin/ldm** or **/usr/sbin/ldm**.

- **Get configuration of all bound domains**

  The **ldm** command is executed to get the full configuration of all bound domains. See "Listing configuration of bound domains" on the next page.

- **Get general networking configuration**

  Standard networking discovery is performed, which involves the following commands:

  - netstat
  - ifconfig
  - dladm

  For more information see "UNIX-Based Processes" on page 935.

- **Get names of interfaces that were created by virtual switches in domain**

  Each virtual switch that is created in the domain, creates additional virtual interfaces (usually named vsw<number>). By bringing these interfaces up, the parent domain can establish connectivity to its switch. To get the names of such interfaces an additional **find** command is run. See "Finding the interfaces created by virtual switches in domains" on page 192.

- **Get number of cores per physical CPU**

  Information about physical cores is taken from the command **/usr/sbin/prtpicl -c other | grep CORE**. After that, the normal approach for CPU discovery on Solaris is followed. For more information, see "How to Discover Host Resources and Applications".

Commands

This section gives examples of the commands used by this discovery.

Obtaining version information of Logical Domains manager

Command

```
/usr/sbin/ldm -V
```

Output

```
Logical Domains Manager (v 2.1)
        Hypervisor control protocol v 1.6
```

```
       Using Hypervisor MD v 1.3

System PROM:
       Hostconfig      v. 1.0.0.       @(#)Hostconfig 1.0.0.b 2010/09/15 03:03
[serpa:release]
       Hypervisor      v. 1.9.0.       @(#)Hypervisor 1.9.0.b 2010/09/15 01:48
       OpenBoot        v. 4.32.0.      @(#)OpenBoot 4.32.0.b 2010/09/29 19:13
```

Listing configuration of bound domains

Command

```
/usr/sbin/ldm list-bindings -p
```

Output

Output is truncated for brevity

```
VERSION 1.5
DOMAIN|name=primary|state=active|flags=normal,control,vio-
service|cons=SP|ncpu=8|mem=4294967296|util=2.4|uptime=10178475
UUID|uuid=11111111-1e91-c63f-99c7-e7484ec50000
MAC|mac-addr=00:21:28:11:73:a0
HOSTID|hostid=0x85117333
CONTROL|failure-policy=ignore
DEPENDENCY|master=
CORE
|cid=0|cpuset=0,1,2,3,4,5,6,7
VCPU
|vid=0|pid=0|util=0.7%|strand=100|cid=0
|vid=1|pid=1|util=0.6%|strand=100|cid=0
|vid=2|pid=2|util=0.9%|strand=100|cid=0
|vid=3|pid=3|util=0.8%|strand=100|cid=0
|vid=4|pid=4|util=2.1%|strand=100|cid=0
|vid=5|pid=5|util=0.5%|strand=100|cid=0
|vid=6|pid=6|util=0.5%|strand=100|cid=0
|vid=7|pid=7|util=3.3%|strand=100|cid=0
MAU
|id=0|cpuset=0,1,2,3,4,5,6,7
MEMORY
|ra=0x8000000|pa=0x8000000|size=4294967296
VARIABLES
|auto-boot?=false
|boot-device=disk0 disk1
|keyboard-layout=US-English
IO
|dev=pci@0|alias=pci
|dev=niu@80|alias=niu
|dev=pci@0/pci@0/pci@8/pci@0/pci@9|alias=MB/RISER0/PCIE0
```

```
|dev=pci@0/pci@0/pci@8/pci@0/pci@1|alias=MB/RISER1/PCIE1
|dev=pci@0/pci@0/pci@9|alias=MB/RISER2/PCIE2
|dev=pci@0/pci@0/pci@8/pci@0/pci@a|alias=MB/RISER0/PCIE3
|dev=pci@0/pci@0/pci@8/pci@0/pci@2|alias=MB/RISER1/PCIE4
|dev=pci@0/pci@0/pci@8/pci@0/pci@8|alias=MB/RISER2/PCIE5
|dev=pci@0/pci@0/pci@1/pci@0/pci@2|alias=MB/NET0
|dev=pci@0/pci@0/pci@1/pci@0/pci@3|alias=MB/NET2
|dev=pci@0/pci@0/pci@2|alias=MB/SASHBA
VCC|name=vcc|port-range=5001-5010
|client=guest1@vcc|port=5001
VSW|name=vsw1|mac-addr=00:21:28:11:73:a2|net-dev=e1000g2|dev=switch@1|default-vlan-
id=1|pvid=1|vid=|mode=|mtu=1500|linkprop=|id=1
|peer=vnet0@guest1|mac-addr=00:14:4f:f9:6f:4d|pvid=1|vid=|mtu=1500
VDS|name=vds0
|vol=guest1os|opts=|dev=/dev/zvol/dsk/ldoms/guest1os|mpgroup=
|vol=guest1ap|opts=|dev=/dev/zvol/dsk/ldoms/guest1ap|mpgroup=
|vol=L1_2234|opts=|dev=/dev/dsk/c6t60060480000290101177533032323334d0s2|mpgroup=
|vol=L1_2228|opts=|dev=/dev/dsk/c6t60060480000290101177533032323238d0s2|mpgroup=
|vol=L1_221C|opts=|dev=/dev/dsk/c6t60060480000290101177533032323143d0s2|mpgroup=
|client=vdisk0@guest1|vol=guest1os
|client=vdisk1@guest1|vol=guest1ap
|client=vdisk2@guest1|vol=L1_2234
|client=vdisk3@guest1|vol=L1_2228
|client=vdisk4@guest1|vol=L1_221C
VCONS|type=SP
DOMAIN|name=guest1|state=active|flags=normal|cons=5001|ncpu=32|
mem=19327352832|util=0.0|uptime=8584562
UUID|uuid=22222222-8dfb-6742-9705-d2f4d4310000
MAC|mac-addr=00:14:4f:f9:35:8f
HOSTID|hostid=0x84f93555
CONTROL|failure-policy=ignore
DEPENDENCY|master=
CORE
|cid=1|cpuset=8,9,10,11,12,13,14,15
|cid=2|cpuset=16,17,18,19,20,21,22,23
|cid=3|cpuset=24,25,26,27,28,29,30,31
|cid=4|cpuset=32,33,34,35,36,37,38,39
VCPU
|vid=0|pid=8|util=0.3%|strand=100|cid=1
|vid=1|pid=9|util=0.1%|strand=100|cid=1
|vid=2|pid=10|util=0.0%|strand=100|cid=1
|vid=3|pid=11|util=0.0%|strand=100|cid=1
|vid=4|pid=12|util=0.3%|strand=100|cid=1
|vid=5|pid=13|util=0.0%|strand=100|cid=1
|vid=6|pid=14|util=0.0%|strand=100|cid=1
|vid=7|pid=15|util=0.0%|strand=100|cid=1
|vid=8|pid=16|util=0.0%|strand=100|cid=2
|vid=9|pid=17|util=0.0%|strand=100|cid=2
```

```
|vid=10|pid=18|util=0.0%|strand=100|cid=2
|vid=11|pid=19|util=0.0%|strand=100|cid=2
|vid=12|pid=20|util=0.0%|strand=100|cid=2
|vid=13|pid=21|util=0.0%|strand=100|cid=2
|vid=14|pid=22|util=0.3%|strand=100|cid=2
|vid=15|pid=23|util=0.1%|strand=100|cid=2
|vid=16|pid=24|util=0.0%|strand=100|cid=3
|vid=17|pid=25|util=0.0%|strand=100|cid=3
|vid=18|pid=26|util=0.1%|strand=100|cid=3
|vid=19|pid=27|util=0.1%|strand=100|cid=3
|vid=20|pid=28|util=0.0%|strand=100|cid=3
|vid=21|pid=29|util=0.0%|strand=100|cid=3
|vid=22|pid=30|util=0.0%|strand=100|cid=3
|vid=23|pid=31|util=0.0%|strand=100|cid=3
|vid=24|pid=32|util=3.6%|strand=100|cid=4
|vid=25|pid=33|util=0.0%|strand=100|cid=4
|vid=26|pid=34|util=0.0%|strand=100|cid=4
|vid=27|pid=35|util=0.0%|strand=100|cid=4
|vid=28|pid=36|util=0.2%|strand=100|cid=4
|vid=29|pid=37|util=0.0%|strand=100|cid=4
|vid=30|pid=38|util=0.0%|strand=100|cid=4
|vid=31|pid=39|util=0.0%|strand=100|cid=4
MAU
|id=1|cpuset=8,9,10,11,12,13,14,15
|id=2|cpuset=16,17,18,19,20,21,22,23
|id=3|cpuset=24,25,26,27,28,29,30,31
|id=4|cpuset=32,33,34,35,36,37,38,39
MEMORY
|ra=0x8000000|pa=0x108000000|size=19327352832
VARIABLES
|boot-device=/virtual-devices@100/channel-devices@200/disk@0:a disk net
|keyboard-layout=US-English
VNET|name=vnet0|dev=network@0|service=vsw1@primary|mac-
addr=00:14:4f:f9:6f:4d|mode=|pvid=1|vid=|mtu=1500|linkprop=|id=0
|peer=vsw1@primary|mac-addr=00:21:28:11:73:a2|mode=|pvid=1|vid=|mtu=1500
VDISK|name=vdisk0|vol=guest1os@vds0|timeout=|dev=disk@0|
server=primary|mpgroup=|id=0
VDISK|name=vdisk1|vol=guest1ap@vds0|timeout=|dev=disk@1|
server=primary|mpgroup=|id=1
VDISK|name=vdisk2|vol=L1_2234@vds0|timeout=|dev=disk@2|
server=primary|mpgroup=|id=2
VDISK|name=vdisk3|vol=L1_2228@vds0|timeout=|dev=disk@3|
server=primary|mpgroup=|id=3
VDISK|name=vdisk4|vol=L1_221C@vds0|timeout=|dev=disk@4|
server=primary|mpgroup=|id=4
VCONS|group=guest1|service=vcc@primary|port=5001
```

Finding the interfaces created by virtual switches in domains

Command

```
find /devices/virtual-devices@100 -type c -name virtual-network-switch*
```

Output

```
/devices/virtual-devices@100/channel-devices@200/virtual-network-switch@0:vsw0
/devices/virtual-devices@100/channel-devices@200/virtual-network-switch@1:vsw1
```

Finding the number of cores per CPU

Command

```
/usr/sbin/prtpicl -c other | grep CORE
```

Output

```
CORE0 (other, b2333315a2)
CORE1 (other, b2333315cc)
CORE2 (other, b2333315f6)
CORE3 (other, b233331620)
CORE0 (other, b2333315a2)
CORE1 (other, b2333315cc)
CORE2 (other, b2333315f6)
CORE3 (other, b233331620)
```

The output shows two physical CPUs with 4 cores each.

Troubleshooting and Limitations

- Due to the technical limitation and architecture of LDOMs, not all guest domains can be reported by the discovery job. Guest domains that have no network connectivity to the Virtual Switch located in this control domain cannot be reported, since there is not enough identification information for such a domain.

- Several virtual network devices created by LDOMs have MAC addresses assigned. These MACs can be autogenerated or manually assigned. In some cases, different LDOM servers generate the MACs. Since there is no other identification information about guest domains available besides the MAC addresses of their virtual interfaces, if MACs on different LDOMs match, the corresponding Nodes of the domains may also merge in CMDB.

Chapter 13: Oracle VM for x86 Discovery

This chapter includes:

Overview

Oracle VM is a platform that provides a fully equipped environment with all the latest benefits of virtualization technology. Oracle VM enables deployment of operating systems and application software within a supported virtualization environment. Oracle VM insulates users and administrators from the underlying virtualization technology and allows daily operations to be conducted using goal-oriented GUI interfaces.

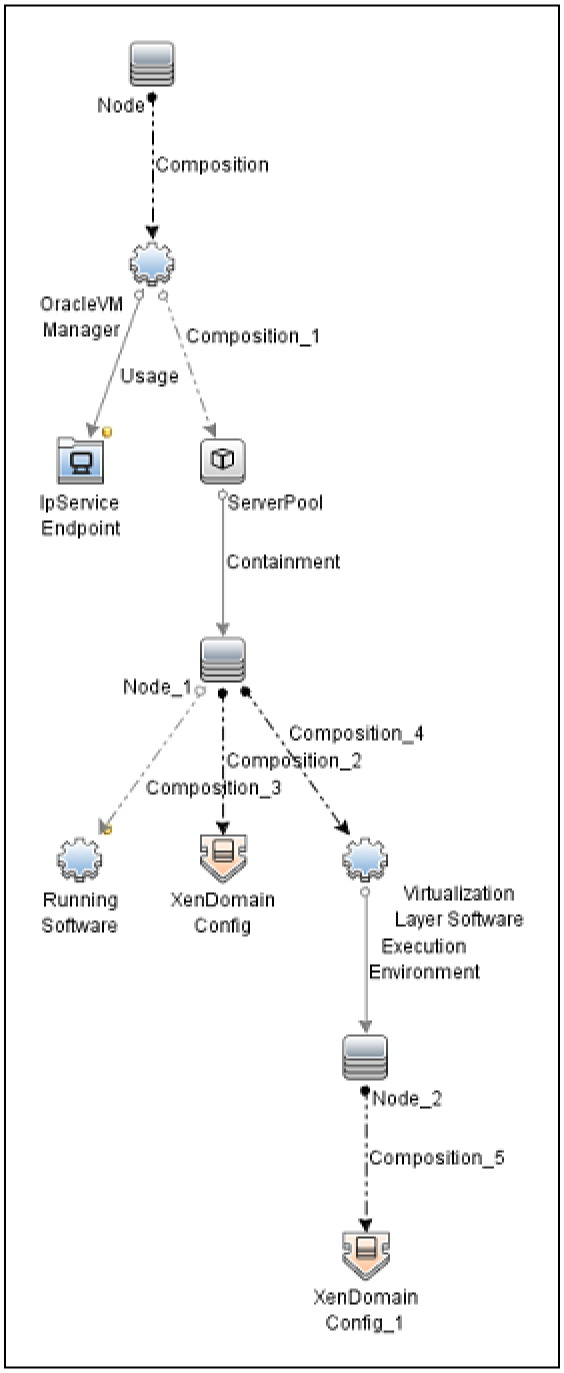
Oracle VM is an enterprise class server virtualization solution comprised of Oracle VM Server for x86, Oracle VM Server for SPARC, and Oracle VM Manager. For x86 servers, Oracle VM includes Oracle VM Manager and Oracle VM Server for x86.

Supported Versions

Oracle VM for x86 Discovery supports Oracle VM Server for x86, version 3.2.1 and later versions.

Topology

The topology for Oracle VM for x86 discovery is shown below.



How To Discover Oracle VM for x86 Topology

This section describes how to discover the Oracle VM for x86 Topology.

Prerequisites

- Ensure that there is shell connectivity to the Oracle VM Manager
- Set up SSH protocol credentials. If a port is specified in the credentials, then credentials are applied only to destinations where the port is listening. Otherwise, credentials are skipped.

Run the Discovery

Note: Oracle VM for x86 Topology Discovery can be performed in shallow or deep mode. To perform a shallow discovery, carry out steps 1-3 below. To perform a deep discovery, carry out all the steps below.

1. Run the **Range IPs by ICMP** job (discovers reachable IPs).
2. Run the **Host Connection by Shell** job (discovers the target host and shell connectivity to the host).
3. Run the **Host Applications by Shell** job (discovers Oracle VM agents, the Oracle VM Manager, and the Oracle VM CLI listening port).

   Note: If you want to perform only a shallow discovery, stop here and do not continue to the next step.

4. Run the **Oracle VM for x86 by Manager Main CLI** job (discovers virtualization topology managed by the Oracle VM Manager).

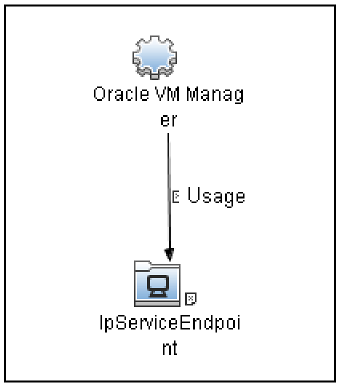
Oracle VM for x86 by Manager Main CLI Job

Adapter

ID: oracle_vm_manager_by_maincli

Trigger TQL

- Trigger CI: **OracleVmManager**, with the optional endpoint **ovm_manager_cli_ssh** connected.
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
IpServiceEndpoint	ServiceNames contains ovm_manager_cli_ssh.

Discovery Flow

This job iterates over available SSH credentials and attempts to connect against the manager destination with the port specified in one of the following places:

- Main CLI endpoint port
- SSH credentials port
- Default port: 10000 (specified in script)

If a connection is established, Discovery is performed using OVM CLI in the following order:

1. This job sets the CLI command output to XML (command: **set output=xml**).
2. This job lists all the available servers (command: **list Server**).
3. This job lists all the virtual machines (command: **list VM**).

4. For each server, this job obtains its details (command: **show Server**).

5. For each virtual machine, this job obtains its details (command: **show Vm**).

6. This job gets the version of the CLI/Oracle VM Manager (command: **showversion**).

Oracle VM Manager Discovery by Main CLI Adapter

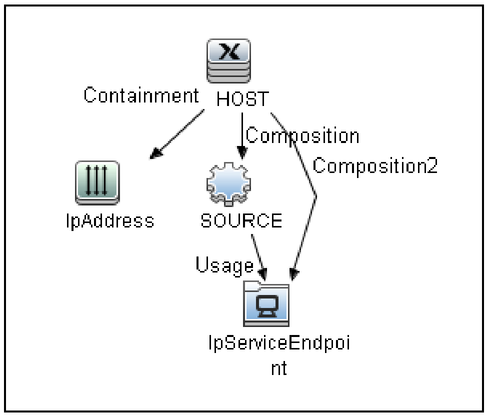
ID

`oracle_vm_manager_by_maincli`

Input CIT

`OracleVmManager`

Input TQL



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id:NA}
ip_address	${SOURCE.application_ip:NA}
protocol_port	${IpServiceEndpoint.network_port_number:NA}

Used Scripts

- **manager_by_ovm_cli.py.** The entry point script.

Discovered CITs

- Composition
- Containment
- Dependency
- ExecutionEnvironment
- Interface
- IpServiceEndpoint
- Node
- OracleVMManager
- OracleVmServerPool
- Realization
- RunningSoftware
- Usage

- Virtualization Layer Software
- VM Server
- XenDomainConfig (xen_domain_config) for VM Server and Virtual Machine

Parameters

- **commandExecutionDurationInMs.** The time (ms) allocated for execution of all CLI commands. The specified time value depends on the load factor of the manager host. The default value is 2000 ms (2 seconds).
- **reportStoppedVMs.** Performs discovery of stopped VMs. The default value is **false**.

Chapter 14: Solaris Zones Discovery

This chapter includes:

Overview

The Solaris Zones partitioning technology is used to virtualize operating system services and provide an isolated and secure environment for running applications. A zone is a virtualized operating system environment created within a single instance of the Solaris Operating System. When you create a zone, you produce an application execution environment in which processes are isolated from the rest of the system. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with superuser credentials cannot view or affect activity in other zones.

A zone also provides an abstract layer that separates applications from the physical attributes of the machine on which they are deployed. Examples of these attributes include physical device paths.

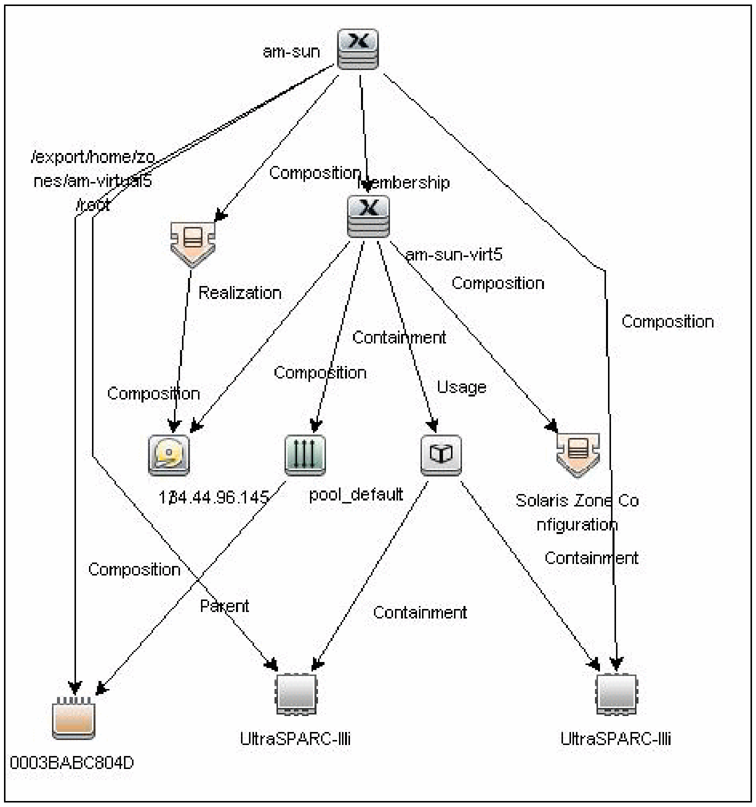
Supported Versions

Solaris Zones discovery supports Solaris 10 or later.

Topology

The following image displays the topology of the Solaris Zones discovery with sample output:

Note: For a list of discovered CITs, see "Solaris Zones by TTY Job" on page 205.



How to Discover Solaris Zones

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the SSH and Telnet protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Set up permissions**

   Zones are discovered from the Global Zone of the machine, so you should have appropriate permissions to:

   - access the Global Zone and perform discovery
   - log into the Non-global Zones through the **zlogin** command

   Note: The zlogin command can be executed:

   i. With root user (the default value)

   ii. With a connection to the global zone user. You can configure this option with the discovery pattern parameter **zloginWithConnectedUser**.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   b. Run the **Host Connection by Shell** job to discover Shell connectivity and basic information about the hosts.

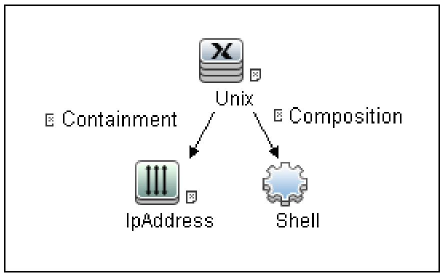
   c. Run the **Solaris Zones by TTY** job to discover zone configuration.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Solaris Zones by TTY Job

- "Solaris Zones by TTY Job" above
- "Solaris Zones by TTY Job" above
- "Solaris Zones by TTY Job" above
- "Solaris Zones by TTY Job" above
- "Solaris Zones by TTY Job" above
- "Solaris Zones by TTY Job" above

Trigger Query

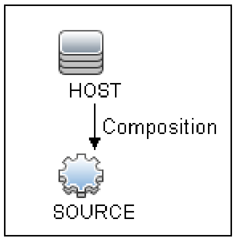


Adapter

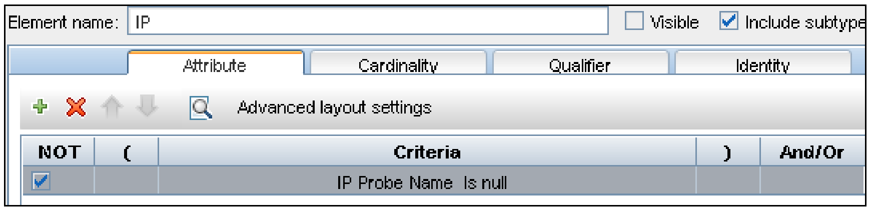
The Solaris Zones by TTY Job uses the **SolarisZone_Disc_By_TTY** adapter.

- **Input Query**

  The Input query contains one Shell CI only:



- **IP Process**



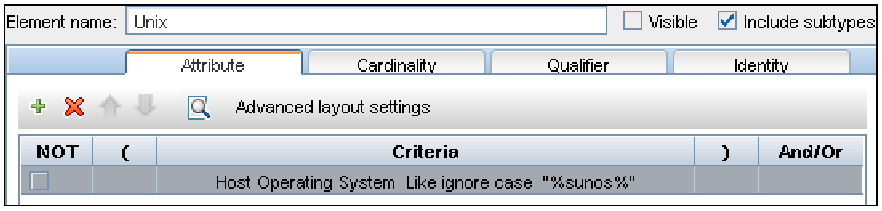
Element name: IP ☐ Visible ☑ Include subtype

Attribute | Cardinality | Qualifier | Identity

Advanced layout settings

NOT	(	Criteria	)	And/Or
☑		IP Probe Name Is null		

- **UNIX Process**



Element name: Unix ☐ Visible ☑ Include subtypes

Attribute | Cardinality | Qualifier | Identity

Advanced layout settings

NOT	(	Criteria	)	And/Or
☐		Host Operating System Like ignore case "%sunos%"		

Parameters

Parameter	Description
zloginWithConnectedUser	If **true**, zlogin is executed with a connection to the global user account. If **false**, zlogin uses the root account. **Default:** False.

Created/Changed Entities

- **Additional CI Types**:
  - Solaris Zones Config
  - Solaris Resource Pool
- **Additional valid links**:
  - Solaris Resource Pool > **Containment** > CPU
  - Unix > **Usage** > Solaris Resource Pool
  - Unix > **Composition** > Solaris Resource Pool
- **Modified views:**

    - Solaris Zones view

- **Modified scripts**:
    - SolarisZone_Disc_By_TTY.py
- **Additional enrichments**:
    - Solaris Zones Networking

Discovered CITs

- Composition
- Containment
- Cpu
- Fibre Channel HBA
- FileSystem
- FileSystemExport
- IPMP Group
- Interface
- IpAddress
- IpSubnet
- Membership
- Node
- Parent
- Realization
- Solaris Resource Pool
- Solaris Zone Config
- Usage

Note: To view the topology, see "Topology" on page 203.

Discovery Mechanism

This section includes the following commands:

- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205
- "Solaris Zones by TTY Job" on page 205

Verify the Connected OS is Zone-compliant

Command	uname -r
Example of output	`5.10`
Values taken	5.10
Comments	This command retrieves the Solaris OS version. If it is 5.10 it is assumed that the version supports zones and discovery continues. If it is not equal to 5.10 (for example, 5.9) it is assumed the host is not zone-compliant and discovery ends with the message `Server does not support zones`.

Obtain List of Zones, Verify the Connected Host is Global Zone

Command	/usr/sbin/zoneadm list -cp

Example of output 1	`0:global:running:/::native:shared`
	
	`27:zone1:running:/var/opt/zones/zone1 :11559a59-3c6f-6a6e-a723-cc8159351247: native:excl`
	
	`-:zone2:configured:/var/opt/zones/ zone2::native:shared`

Example of output 2 (no root permissions)	`0:global:running:/`
	
	`1:am-virtual6:running:/export/home/ zones/am-virtual6`
	
	`5:am-virtual5:running:/export/home/ zones/am-virtual5`
	
	`7:am-virtual3:running:/virtual/3`
	
	`9:am-virtual1:running:/am-virtual/1`

Values taken	Name of the zone: zone1 Status of the zone: running Zone path: /var/opt/zones/zone1
Comments	This command gives the list of zones and their configuration including names, status, and path. The following is verified: • That **global** is present in the output. If it is missing, the zone that discovery connected to is not global. • There is at least one more non-global zone apart from the global zone. If this is not true, discovery ends with the message **`Server does not have zones defined`**.

Obtain Configuration for Each of the Non-global Zones

Command	/usr/sbin/zonecfg -z <zonename> info

Example of output 1	
	```
	zonename: zone1
	zonepath: /var/opt/zones/zone1
	brand: native
	autoboot: true
	bootargs: -m verbose
	pool:
	limitpriv: default,sys_time
	scheduling-class:
	ip-type: exclusive
	fs:
	dir: /mnt/globalzone
	special: /var/opt/zone1-data
	raw not specified
	type: lofs
	options: []
	net:
	address not specified
	physical: bge2
	defrouter not specified
	device
	match: /dev/bge2
	dedicated-cpu:
	ncpus: 1
	importance: 1
	capped-cpu:
	[ncpus: 1.00]
	capped-memory:
	physical: 16G
	[swap: 8G]
	[locked: 12G]
	```

Example of output 2

```
zonename: zone2
zonepath: /var/opt/zones/zone2
brand: native
autoboot: true
bootargs: -m verbose
pool:
limitpriv: default
scheduling-class: FSS
ip-type: shared
fs:
dir: /mnt
special: /var/opt/zone2-data
raw not specified
type: lofs
options: []
net:
address: 134.44.0.100
physical: bge0
defrouter not specified
device
match: /dev/pts*
rctl:
name: zone.cpu-shares
value: (priv=privileged,limit=5,action=none)
```

Values taken	The following information is obtained from the output: • brand (if it is not specified it is assumes to be native) • autoboot • resource pool name • limit privileges • scheduling class • ip type • all mounted file systems • networking information (IP and/or network interface) • dedicated CPUs and their importance • memory caps • cpu caps • cpu shares
Comments	This command is run for each non-global zone found. Most of these properties are stored in the **Solaris Zone Config** CI. File systems are reported as a File System Export from global zone to non-global. The resource pool name is used to create a link to a corresponding resource pool CI.

Obtain MAC Addresses for Interfaces of Global Zone

Command	/usr/bin/netstat -np

Example of output	
	```
	Net to Media Table: IPv4
	Device IP Address Mask Flags Phys Addr
	------ -------------------- --------------- -------- ---------------
	bge0 134.44.0.101 255.255.255.255 o 00:15:f2:05:9e:ff
	bge0 134.44.1.150 255.255.255.255 o 00:15:f2:9b:2d:96
	bge0 134.44.0.100 255.255.255.255 SPLA 00:14:4f:82:74:a4
	bge0 134.44.98.135 255.255.255.255 o 00:1c:c0:2b:57:35
	bge0 224.0.0.0 240.0.0.0 SM 01:00:5e:00:00:00
	```

Values taken	MAC addresses of corresponding interfaces.
Comments	This command retrieves the list of all interfaces except for the dedicated interface used in exclusive zones. Interfaces in the global zone are shared with shared zones, so this command runs only once. MAC addresses and information in the zonecfg output enables the creation of shared non-global zone Host CIs.

Obtain IP Information for Global Zone

Command	/usr/sbin/ifconfig -a

Example of output	
	```
	lo0: flags=2001000849<UP,LOOPBACK,
	RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
	inet 127.0.0.1 netmask ff000000
	lo0:1: flags=2001000849<UP,LOOPBACK,RUNNING,
	MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
	zone zone2
	inet 127.0.0.1 netmask ff000000
	e1000g1: flags=1000843<UP,BROADCAST,RUNNING,
	MULTICAST,IPv4> mtu 1500 index 2
	inet 134.44.0.50 netmask ffffff00
	 broadcast 134.44.0.255
	e1000g1:1: flags=1000843<UP,BROADCAST,RUNNING,
	MULTICAST,IPv4> mtu 1500 index 2
	zone zone2
	inet 134.44.0.100 netmask ffffff00
	 broadcast 134.44.0.255
	```

Values taken	The MAC addresses of corresponding interfaces.
Comments	This command retrieves the IP configuration for the global zone that is shared with corresponding shared non-global zones. This information is used to report IP addresses and link them to corresponding network interfaces.

Obtain IP Information of Exclusive Zones

Command	/usr/sbin/zlogin -l <username> <zonename> /usr/sbin/ifconfig -a

Example of output	`lo0: flags=2001000849<UP,LOOPBACK,RUNNING,`
	`MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1`
	`inet 127.0.0.1 netmask ff000000`
	`bge2: flags=201004843<UP,BROADCAST,RUNNING,`
	`MULTICAST,DHCP,IPv4,CoS> mtu 1500 index 2`
	`inet 134.44.0.200 netmask fffffc00`
	`broadcast 134.44.0.255`
	`ether 0:14:4f:82:74:a6`

Values taken	All IPs that are present except loopback.
Comments	This command retrieves the IP information for exclusive non-global zones. The -l <user> switch is added to simplify setting up the sudo pattern for zlogin, but it can be removed from the job parameters. **Note**: Discovery runs zlogin for zones in a running state only.

Obtain MAC Addresses for Dedicated Interfaces of Exclusive Zones

Command	/usr/sbin/zlogin -l <username> <zonename> /usr/bin/netstat -np

Example of output	`Net to Media Table: IPv4`
	
	`Device IP Address Mask Flags Phys Addr`
	
	`------ -------------------- --------------- -------- ---------------`
	
	`bge2 134.44.0.200 255.255.255.255 SPLA 00:14:4f:82:74:a6`
	
	`bge2 224.0.0.0 240.0.0.0 SM 01:00:5e:00:00:00`

Values taken	MAC addresses.
Comments	MAC addresses of the interfaces are obtained together with interface names. **Note:** Discovery runs zlogin for zones in a running state only.

Obtain CPU Information in Global Zone

Command	/usr/sbin/psrinfo -v

Example of output	`Status of virtual processor 0 as of: 05/03/2010 16:00:15`
	` on-line since 04/26/2010 19:45:40.`
	` The sparcv9 processor operates at 1200 MHz,`
	` and has a sparcv9 floating point processor.`
	`Status of virtual processor 1 as of: 05/03/2010 16:00:15`
	` on-line since 04/26/2010 19:45:42.`
	` The sparcv9 processor operates at 1200 MHz,`
	` and has a sparcv9 floating point processor.`

Values taken	Number of virtual CPUs with IDs Virtual processor names (sparcv9) Processors speeds (1200)
Comments	For each instance of the virtual processor, discovery creates a CPU with a name (sparcv9) and speed (1200). They are linked to the global zone. They are also linked to the corresponding resource pool.

Obtain Resource Pools

Command	/usr/sbin/pooladm

Example of output

```
system default
 string system.comment
 int system.version 1
 boolean system.bind-default true
 string system.poold.objectives wt-load
 pool SUNWtmp_zone1
 int pool.sys_id 1
 boolean pool.active true
 boolean pool.default false
 int pool.importance 1
 string pool.comment
 boolean pool.temporary true
 pset SUNWtmp_zone1
pool pool_default
 int pool.sys_id 0
 boolean pool.active true
 boolean pool.default true
 int pool.importance 1
 string pool.scheduler FSS
 string pool.comment
 pset pset_default
```

Example of output *(cont'd)*	`pset SUNWtmp_zone1`
	`int pset.sys_id 1`
	`boolean pset.default false`
	`uint pset.min 1`
	`uint pset.max 1`
	`string pset.units population`
	`uint pset.load 0`
	`uint pset.size 1`
	`string pset.comment`
	`boolean pset.temporary true`
	`cpu`
	`int cpu.sys_id 0`
	`string cpu.comment`
	`string cpu.status on-line`

Values taken	• Pools: ▪ Name ▪ Is default ▪ Is active ▪ Importance ▪ Scheduler • Pset: ▪ Name ▪ Min CPUs ▪ Max CPUs ▪ Objectives Relations from **Pool** to **Pset** and from **Pset** to assigned CPUs by IDs

Comments	This information enables reporting pools and links them to corresponding CPUs of the global zone by IDs. Currently discovery reports pool and its pset as one entity. If the resource pools facility is not used or not active discovery cannot read the configuration, but still reports the default (dummy) pool without attributes; all CPUs are linked there. If the non-global zone includes the name of the pool in the configuration discovery links the zone to this pool. If the non-global zone has a dedicated-cpu property set, discovery calculates the name of the temporary dynamic pool for linkage. The name takes the following format: SUNWtmp_<zonename>.

Obtain Fibre Channel Adapters

Command	/usr/sbin/fcinfo hba-port

Example of output	
	```
	HBA Port WWN: 2100001c3491b18a
	OS Device Name: /dev/cfg/c1
	Manufacturer: QLogic Corp.
	Model: 555-1156-02
	Firmware Version: 05.01.00
	FCode/BIOS Version: BIOS: 2.2;
	  fcode: 2.1; EFI: 2.0;
	Serial Number: 0708R00-4259732555
	Driver Name: qlc
	Driver Version: 20090610-3.21
	Type: N-port
	State: online
	Supported Speeds: 1Gb 2Gb 4Gb
	Current Speed: 2Gb
	Node WWN: 2000001c3491b18a
	HBA Port WWN: 2101001c34b1b18a
	OS Device Name: /dev/cfg/c2
	Manufacturer: QLogic Corp.
	Model: 555-1156-02
	Firmware Version: 05.01.00
	FCode/BIOS Version: BIOS: 2.2;
	  fcode: 2.1; EFI: 2.0;
	Serial Number: 0708R00-4259732555
	Driver Name: qlc
	Driver Version: 20090610-3.21
	Type: N-port
	State: online
	Supported Speeds: 1Gb 2Gb 4Gb
	Current Speed: 2Gb
	Node WWN: 2001001c34b1b18a
	```

Values taken	• Port WWN • Os Device Name • Manufacturer • Model • Type • Serial • Driver version
Comments	This information enables discovery to report the Fibre Channel HBA. The OS Device Name is held by the **name** attribute. The Port WWN is held by the **HBA WWN** attribute.

Troubleshooting and Limitations

Problem: The following warning message appears during discovery: `Not enough permissions to execute command, zone is skipped.`

Reason: This might indicate that the script could not retrieve network information for exclusive zones using **zlogin** due to a lack of permissions for the user performing discovery.

Solution:

- Give required permissions to the user.
- Give required permissions to the user.

Chapter 15: VMware Prerequisites

Before running any VMware jobs, you must complete the following prerequisites:

1. **Credentials and Permissions**

   vCenter Server and ESX server require credentials (username and password) for login. In addition, you must have permissions for all entities being discovered to allow retrieval from the server. You should verify that you (and each authorized user) appear in the **Permissions** tab of each entity (host, cluster, virtual machine etc) in the VMware vSphere Client with, at least, a Read-Only role.

2. **Installation of JAR Files**

   You must ensure the appropriate JAR files are installed on the Probe machines. You may do this using Management Zones, or manually.

How to Install the JAR Files Using Management Zones

This task consists of the following steps:

a. Download the **VMware Infrastructure SDK** from:

   http://www.vmware.com/support/developer/vc-sdk/.

   The required version is 5.0 or later.

b. Extract the files to a local folder. For example: C:\VMware-vSphere-SDK.

   You install the jar files through an Inventory Discovery Activity. You may ***either*** create a new Inventory Discovery Activity, or edit an existing one.

c. Create a new Inventory Discovery Activity:

   i. Go to **Data Flow Management > Universal Discovery > Zone Based Discovery > Management Zones**.

   ii. Select the appropriate Management Zone.

   iii. Click the button and select **New Discovery Activity > Inventory**.

   The **New Inventory Discovery Activity** dialog box appears.

iv. Create the activity and activate it by following the online prompts through the activity wizard. On the **Virtualization** page, you must:

A. Select **Include virtualization topology**.

B. Under **Discovery Options**, select **VMware SDK libraries**.

C. Click **Import file** The Import file dialog box appears.

D. You now have two choices: Axis or JAX-WS Libraries. If unsure, select JAX-WS. Select the files to import as follows:

- If you want to use the Axis libraries, go to the **vsphere-ws\java\Axis** folder (for example: C:\VMware-vSphere-SDK\vsphere-ws\java\Axis) and select **vim.jar** and **vim25.jar**.

  > **Note:** The Axis libraries are not included in VMware Infrastructure SDK version 5.1 and later.

- If you want to use the JAX-WS libraries, go to the **vsphere-ws\java\JAXWS\lib** folder (for example: C:\VMware-vSphere-SDK\vsphere-ws\java\JAXWS\lib) and select **vim25.jar**.

To edit an existing Inventory Discovery Activity:

i. Go to **Data Flow Management > Universal Discovery > Zone Based Discovery > Management Zones**.

ii. Select the appropriate Management Zone.

iii. Select the existing Inventory Discovery Activity.

iv. Click the button.

The **Edit Inventory Discovery Activity** dialog box appears.

v. On the **Virtualization** page, complete the steps shown in paragraph 3d above.

How to Manually Install the JAR Files

a. Download the **VMware Infrastructure SDK** from:

http://www.vmware.com/support/developer/vc-sdk/.

The required version is 5.0 or later.

b. Extract the files to a local folder. For example: C:\VMware-vSphere-SDK.

c. Get the jar files from **VMware Infrastructure SDK**. You now have two choices: Axis or JAX-WS Libraries. If unsure, select JAX-WS.

   - If you want to use the Axis libraries, go to the **vsphere-ws\java\Axis** folder (for example: C:\VMware-vSphere-SDK\vsphere-ws\java\Axis) and select **vim.jar** and **vim25.jar**.

     Note: The Axis libraries are not included in VMware Infrastructure SDK version 5.1 and later.

   - If you want to use the JAX-WS libraries, go to the **vsphere-ws\java\JAXWS\lib** folder (for example: C:\VMware-vSphere-SDK\vsphere-ws\java\JAXWS\lib) and select **vim25.jar**.

d. Put the jar files under:

   <probe>/runtime/probeManager/discoveryResources/vmware.

   Note: You must do this for each Probe where VMware discovery is to run.

e. Restart the probe.

Chapter 16: VMware Infrastructure Discovery

This chapter includes:

Supported Versions

Supported Versions for VIM Protocol

Discovery of VMware infrastructure over the VIM protocol is supported by the following servers:

- VirtualCenter 2.5, 2.0
- vCenter Server 4, 4.1, 5.0, and 5.1
- ESX Server 3.0, 3.5, 4.0, 4.1, 5.0, 5.1, and 5.5

Supported Versions for CIM Protocol

Discovery of VMware infrastructure over the CIM protocol is supported by ESX servers 4.x and 5.x.

SSL Support for the VIM Protocol

Web services use http transport which can also be transferred over SSL. The VIM protocol uses SSL by default, but it is possible to configure it without SSL usage.

Each server supporting the VIM protocol (vCenter server or ESX server) has its own SSL certificate. When connecting over SSL you should verify this certificate and accept it:

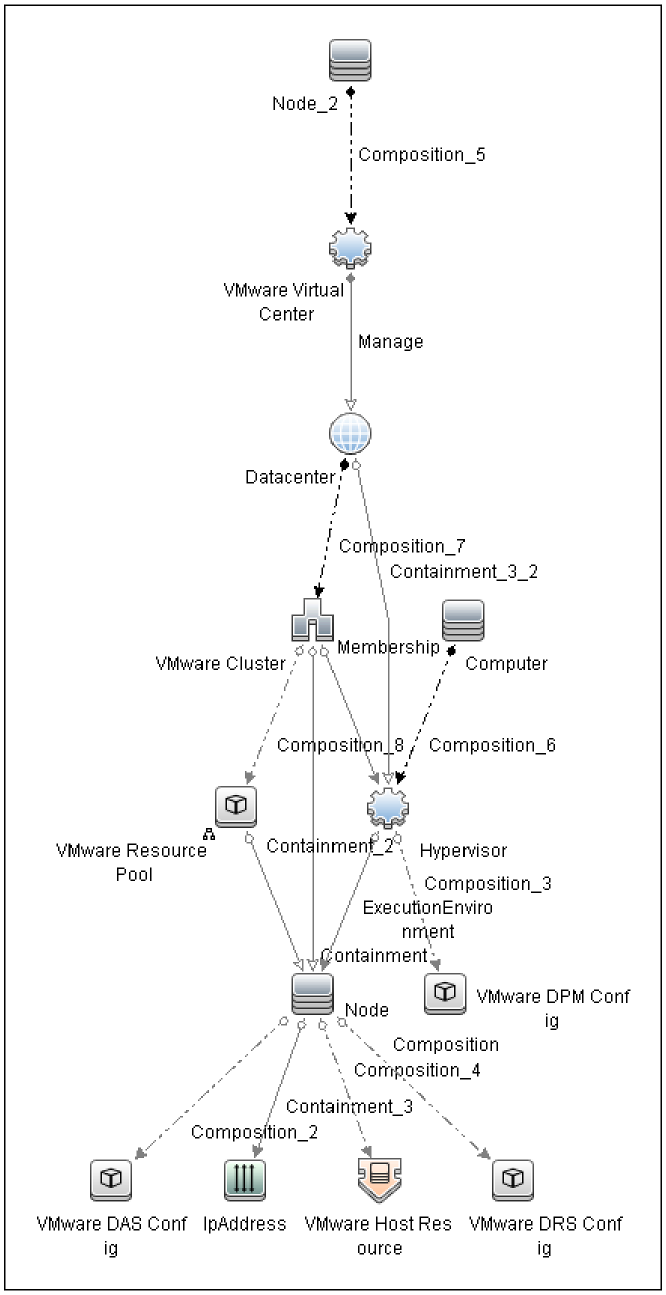
- Import all certificates from the server into a truststore and verify upon each connection while rejecting those that are not present in the set of trusted certificates (this is the secure method).
- Accept all certificates without verification (this is a less secure method).

Currently, DFM supports only one strategy (**accept all certificates always**).

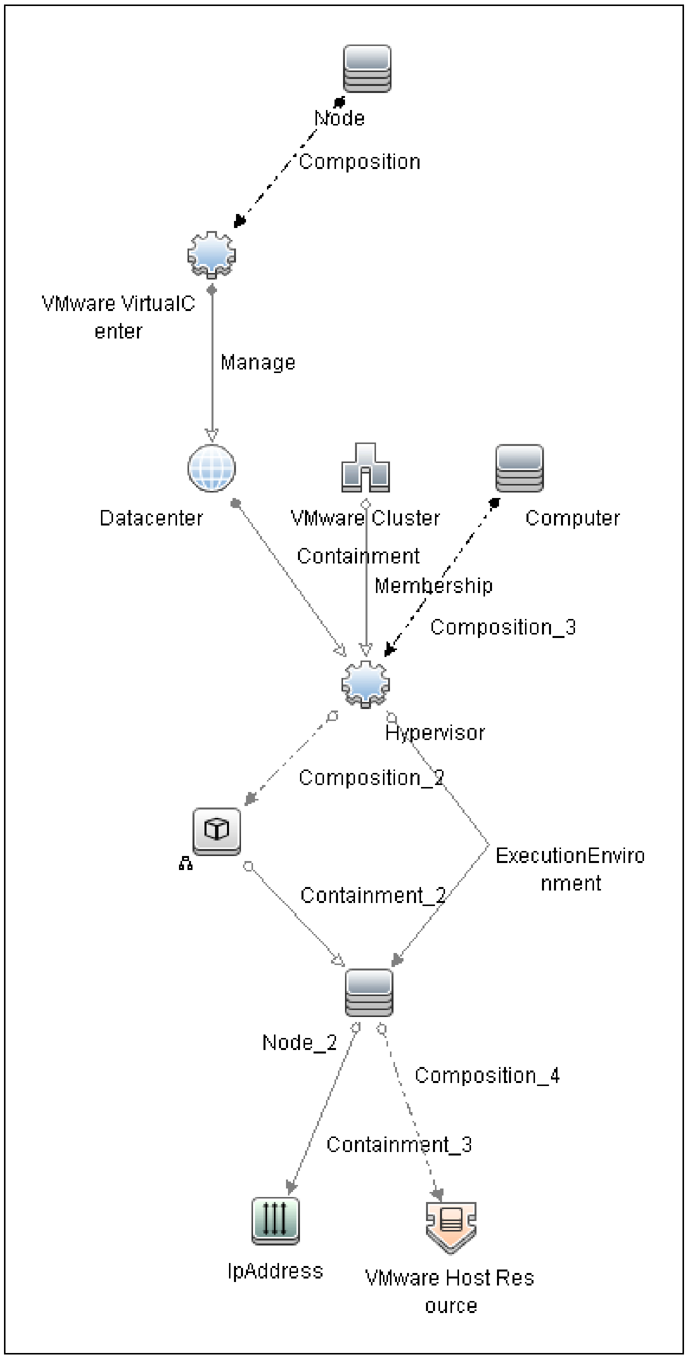
Topology

This section includes:

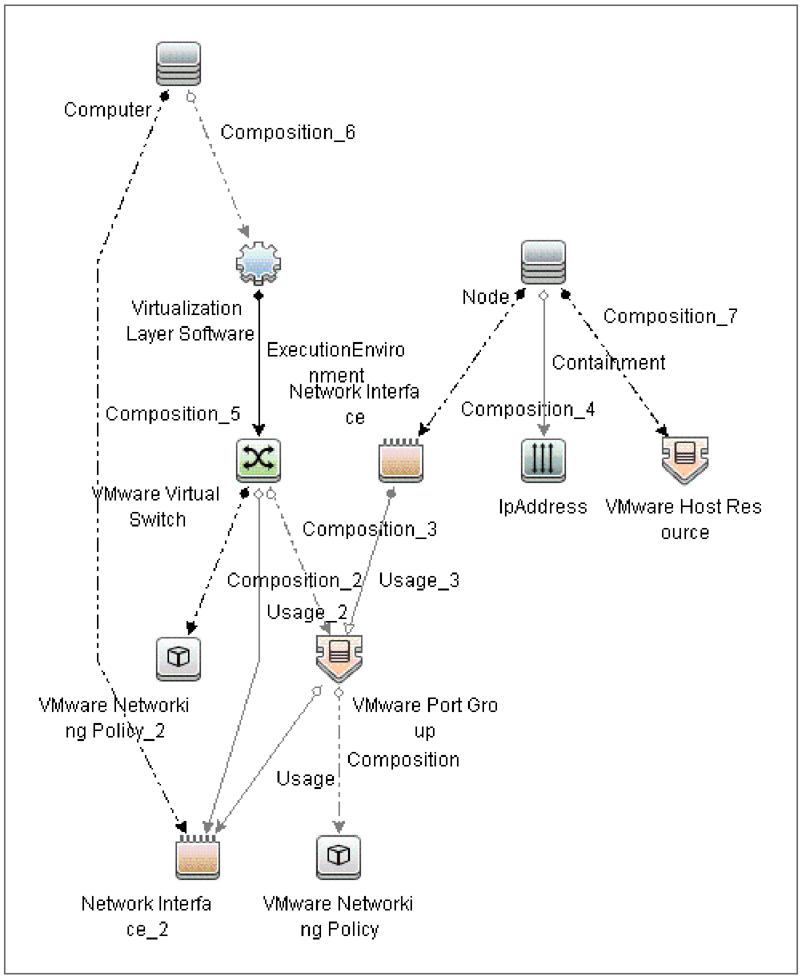
Virtual Topology View for Clusters



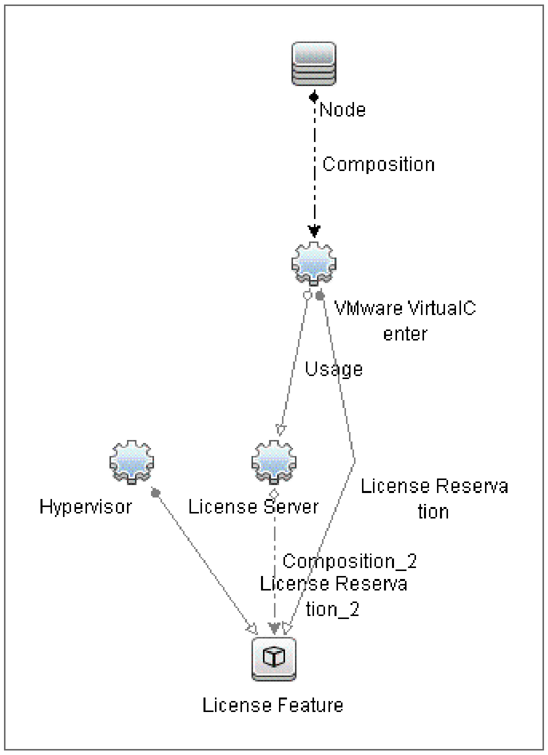
Virtual Topology View for Non-Clusters



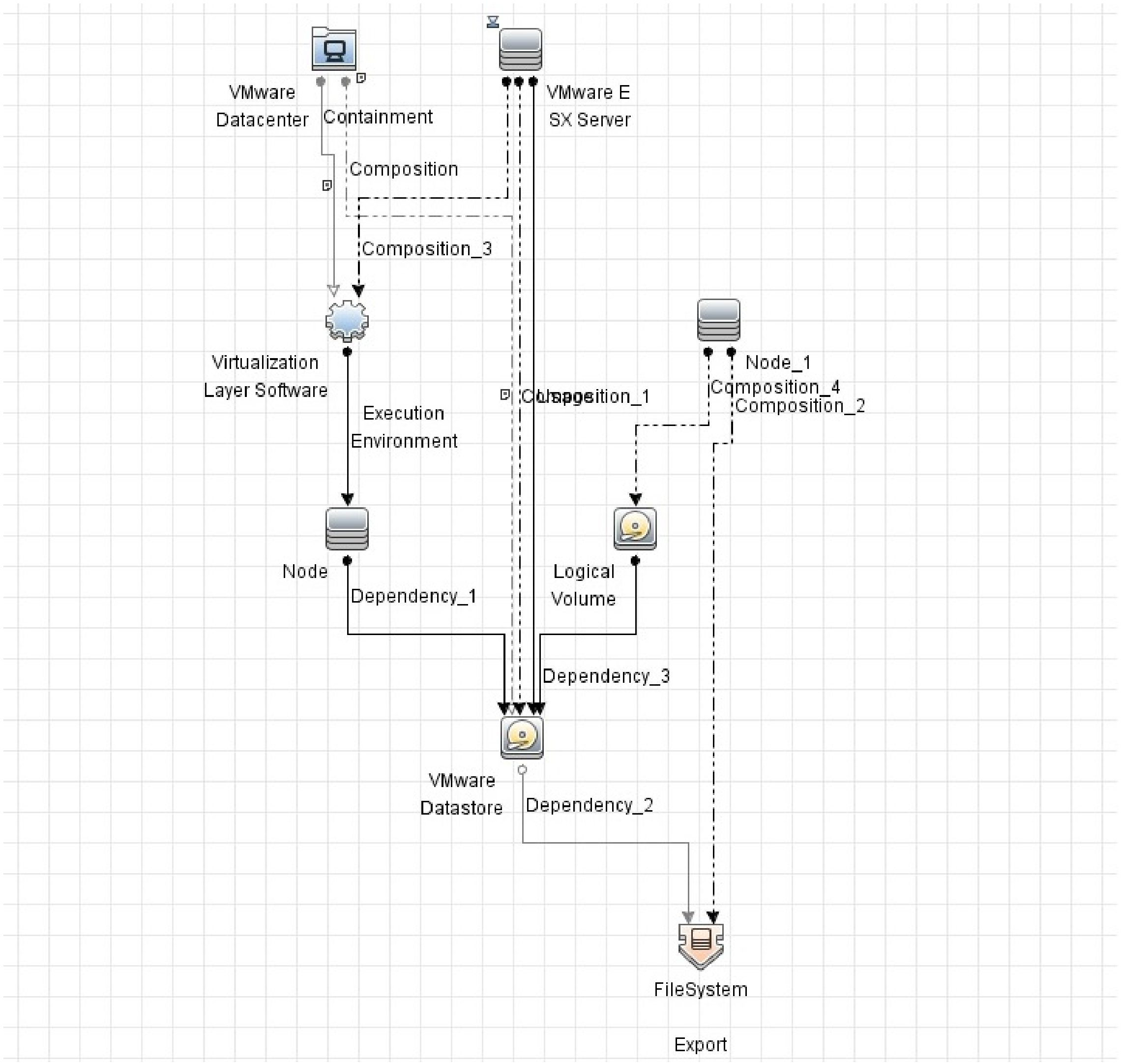
Virtual Topology View for Networking



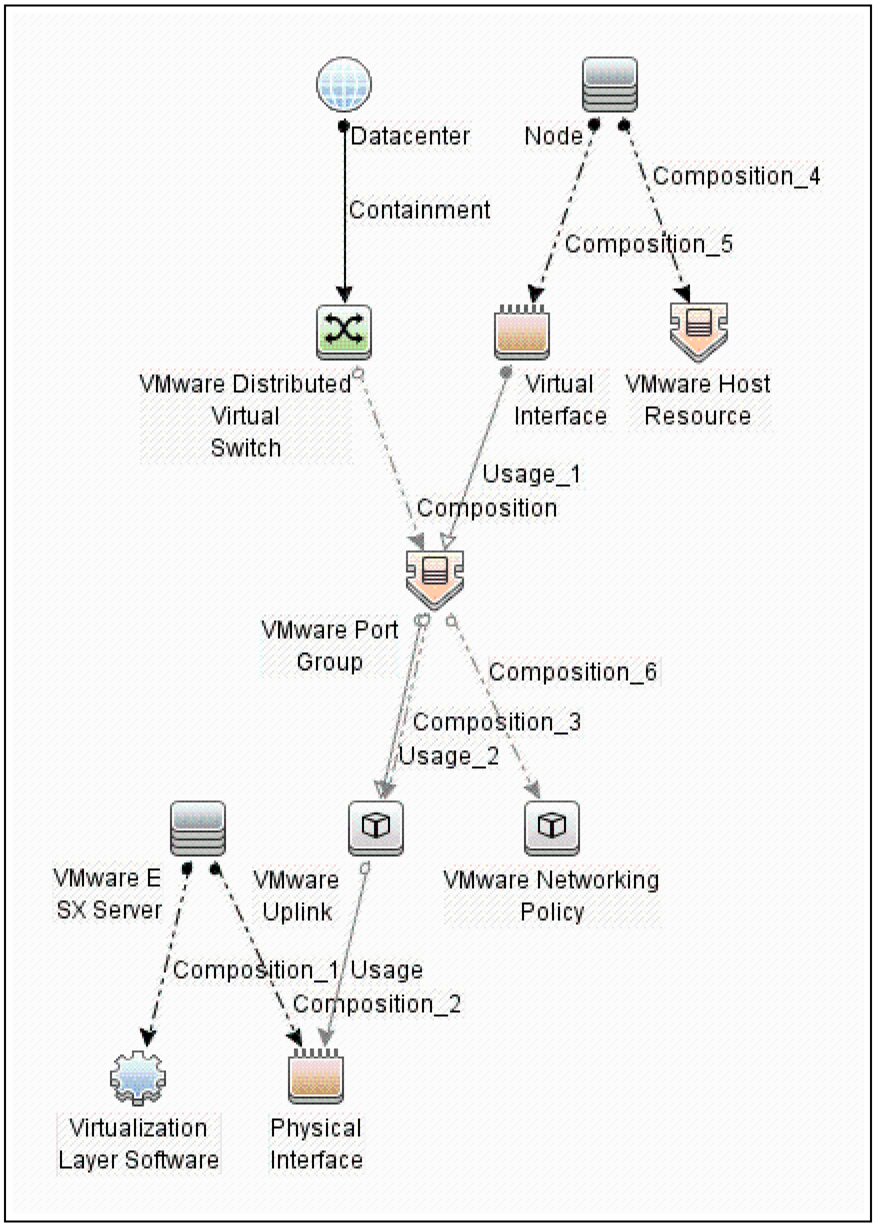
Licensing Topology Map



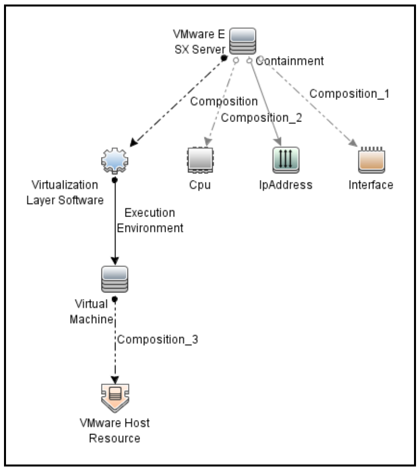
Virtual Topology View for Storage



Topology for Distributed Networking

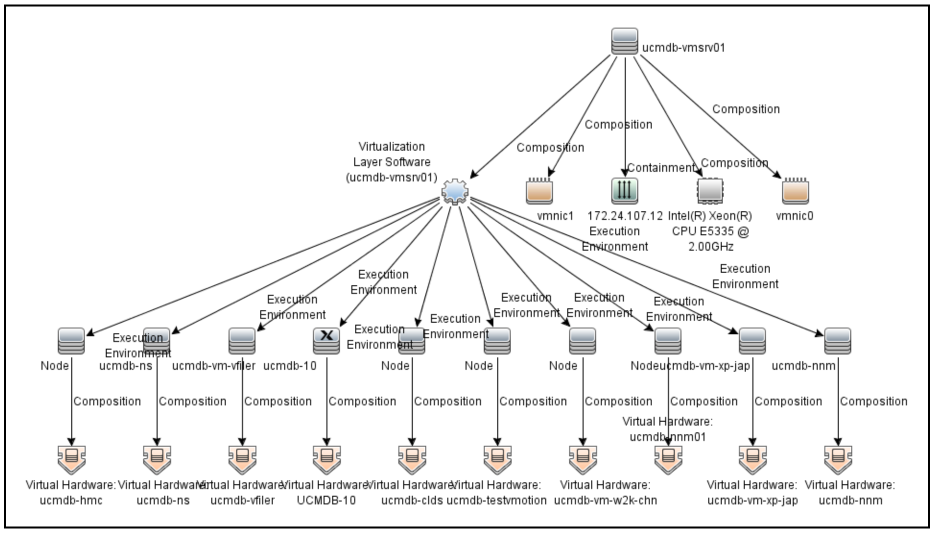


VMware ESX Server Inventory and Virtual Topology View



This view is reported using the CIM protocol.

VMware ESX Server Inventory and Virtual Topology Instances Example


ucmdb-vmsrv01
Virtualization Layer Software (ucmdb-vmsrv01)
Composition
Containment
Execution Environment
vmnic1
172.24.107.12
Intel(R) Xeon(R) CPU E5335 @ 2.00GHz
vmnic0
Node
ucmdb-ns
ucmdb-vm-vfiler
ucmdb-10
ucmdb-vm-xp-jap
ucmdb-nnm
Virtual Hardware: ucmdb-hmc
Virtual Hardware: ucmdb-ns
Virtual Hardware: ucmdb-vfiler
Virtual Hardware: UCMDB-10
Virtual Hardware: ucmdb-clds
Virtual Hardware: ucmdb-testvmotion
Virtual Hardware: ucmdb-vm-w2k-chn
Virtual Hardware: ucmdb-nnm01
Virtual Hardware: ucmdb-vm-xp-jap
Virtual Hardware: ucmdb-nnm

VMware ESX Server Inventory and Virtual Topology Instances are reported using the CIM protocol.

How to Discover VMware VIM Topology

This task describes how to discover the VMware VIM Topology suite of applications. You can discover virtual machines (VM), ESX servers, networking and clustering resources that are running on VMware.

Note: For details on running jobs, see "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

This task includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisites – Set up VMware Infrastructure permissions" below
- "Run Host discovery" on the next page
- "Run Processes discovery" on the next page
- "Run VMware Infrastructure discovery" on the next page

1. **Prerequisite - Set up protocol credentials**

   - The WMI, Shell (Telnet, SSH, NTCMD), and SNMP protocols are required to discover hosts and host processes.

     These protocols require the user name, password, and domain name (the domain name is optional for NTCMD).

   - The VIM protocol is required for all VMware jobs.

     - This protocol requires a user name and password.
     - **Port Number** is optional.
     - **Use SSL.true**: select if the VMware servers are configured to use SSL by default. **false**: select if the VMware servers are configured to use non-secured http.

     For credential information, see "Supported Protocols" in the *HP UCMDB Discovery and Integrations Content Guide*.

2. **Prerequisites – Set up VMware Infrastructure permissions**

The VMware Infrastructure Management (VIM) protocol requires the following permissions:

- **System.Read** permissions for users performing discovery. Users should have permissions for all entities being discovered, and must have been assigned at least a Read-Only role.
- **Global.Licenses** permissions to obtain the total and available number of licenses for each License Feature. If the user does not have these permissions, these attributes remain empty.

3. **Run Host discovery**

   To connect to each potential VMware server (vCenter, VirtualCenter, or ESX), discover its Host CI by running one of the **Host Connection by Shell/WMI** jobs.

4. **Run Processes discovery**

   To connect to each potential VMware server (vCenter, VirtualCenter, or ESX), you must discover Process CIs that match certain criteria, and run Application Signatures discovery by running one of the **Host Applications by Shell/WMI** jobs.

5. **Run VMware Infrastructure discovery**

   The **Virtualization** module includes two jobs for vCenter or VirtualCenter Server discovery and two for ESX Server discovery:

   - If the VMware Infrastructure environment is managed by vCenter or VirtualCenter Servers, run the **VMware vCenter Connection by VIM** job, followed by the **VMware vCenter Topology by VIM** job.
   - If the VMware Infrastructure environment includes unmanaged ESX servers (standalone) or the entire environment is unmanaged, run the **VMware ESX Connection by VIM** job, followed by the **VMware ESX Topology by VIM** job.

   Note: The **Manual VMware VIM Connection** job is intended for use in those instances when the above jobs cannot discover the VMware environment. You must, however, manually run this job. See "How to Run the Manual VMware VIM Connection Job" on page 239.

   For details about each job, see:

- "VMware vCenter Connection by VIM Job" on page 255
- "VMware vCenter Topology by VIM Job" on page 257

- "VMware ESX Connection by VIM Job" on page 245
- "VMware ESX Topology by VIM Job" on page 251

How to Run the Manual VMware VIM Connection Job

You can use this job when the regular connection job (VMware ESX Connection by VIM or VMware vCenter Connection by VIM) cannot run because there is no shell access for the ESX server or the vCenter server.

This task contains the following steps:

1. Go to **Data Flow Management > Universal Discovery > Discovery Modules/Jobs > Discovery Modules > Cloud and Virtualization > Virtualization > VMware**.

2. Select **Manual VMware VIM Connection**.

3. In the **Properties** tab, **Parameters** pane, select **Override** for the **server_url** parameter. Type in **Value** the URL of the target server.

   The format of the URL is: <https or http>://<hostname or IP>:<optional port>/sdk.

   For example: http://secondvcenter:8080/sdk

4. In the **Properties** tab, **Trigger Queries** pane, click the ➕ button. The **Choose Discovery Query** dialog box appears. Select the appropriate probe.

5. Click **OK** to activate the job.

How to Discover VMware ESX Server Topology over CIM

This task describes how to discover the VMware ESX server inventory and virtual topology over CIM.

Prerequisites

- Ensure that CIM agents are enabled on the target ESX servers.
- Define CIM credentials.

This task contains the following steps:

1. To discover the target IPs, run the **Range IPs by ICMP** job.

2. To discover the connectivity to target ESX servers and their basic topology, run the **VMware ESX Connection by CIM** job.

3. To discover the server inventory and virtualization topology, run the **VMware ESX Topology by CIM** job.

Note:

- Discovery uses the CIM client, which is based on SBLIM library (http://sblim.wiki.sourceforge.net).
- Discovery accesses the following three namespaces to collect all required information:
  - root/interop: To verify that the target server implements the Basic Server Profile and to find a reference to the UnitaryComputerSystem instance.
  - root/cimv2: To collect server inventory information.
  - vmware/esxv2: To collect virtualization topology information.

Manual VMware VIM Connection Job

This job supports connection to ESX and vCenter servers.

Adapter

This job uses the VMware_VIM_Connection_Manual adapter.

Parameters

Property	Description
remoteJVMClasspath	The class path used by the external java process. **Note:** Do not change this property from the default.
runInSeparateProcess	When **true**, this enables the execution of the job in the external java virtual machine. **Default:** true. **Note:** Do not change this property from the default.

Discovery Flow

1. The discovery job triggers on the IP address.
2. The discovery obtains the list of credentials applicable to that IP address.
3. The discovery uses each credential and tries to connect to the destination by composing a URL using the above-mentioned IP address, taking the port/protocol from the credential entry.
4. If the connection is successful, it determines whether the connected server is an ESX server or a vCenter server.
5. The job reports the discovered servers

VMware ESX Connection by CIM Job

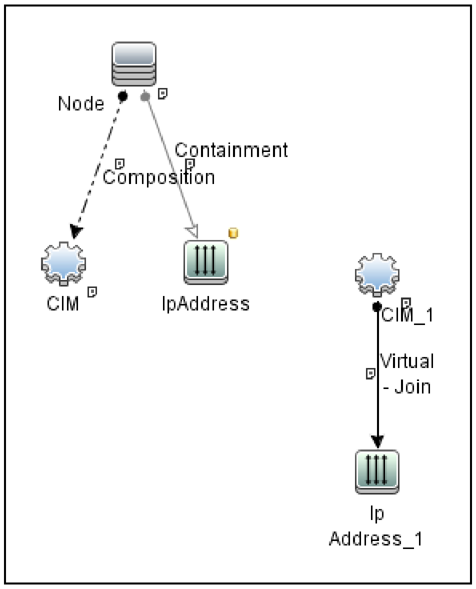
This job discovers the connectivity to target ESX servers and their basic topologies using the CIM protocol.

This section includes:

- "Trigger Query" below
- "Adapter" on the next page
- "Discovered CITs" on the next page

Trigger Query

- Trigger CI: IpAddress
- Trigger Query:



Node Name	Condition
Node	None

Node Name	Condition
CIM	None
IpAddress	`NOT IP Probe Name Is null AND NOT IP Is Broadcast Equal "True"`

Adapter

This job uses the **VMware_ESX_Connection_by_CIM** adapter.

- Triggered CI Data

Name	Value
ip_address	${SOURCE.name}

- Adapter parameters: none
- Scripts:
  - cim.py
  - cim_discover.py
  - host_discoverer.py
  - vmware_cim_discover.py
  - vmware_cim_report.py
  - vmware_esx_connection_by_cim.py

Discovered CITs

- CIM
- Composition
- Containment
- IpAddress
- Interface

- Virtualization Layer Software
- VMware ESX Server

VMware ESX Connection by VIM Job

This job discovers the connections to VMware ESX servers using the VIM protocol.

This section includes:

Discovery Mechanism

Data Flow Management performs the following procedure:

- DFM checks the credentials for the VIM protocol.
- If the current credential includes a defined port, DFM uses this port.

  Otherwise, the port is not specified in the generated connection URL.

  The prefix is determined from the current credential's **use SSL** attribute.
- DFM generates a connection URL: **<prefix>://<ip_address>:<port>/sdk**.
- DFM creates a VMware Infrastructure Client and connects using the generated URL and the user name and password from the credentials.
- If the connection is successful, DFM obtains the product details for the ESX server (version, build, and description), which will be used to populate the attributes of the **Virtualization Layer Software** CI.

  In addition, DFM retrieves the UUID and name of the ESX server. ESX UUID is stored in the `host_key` attribute of the **VMware ESX Server** CI.

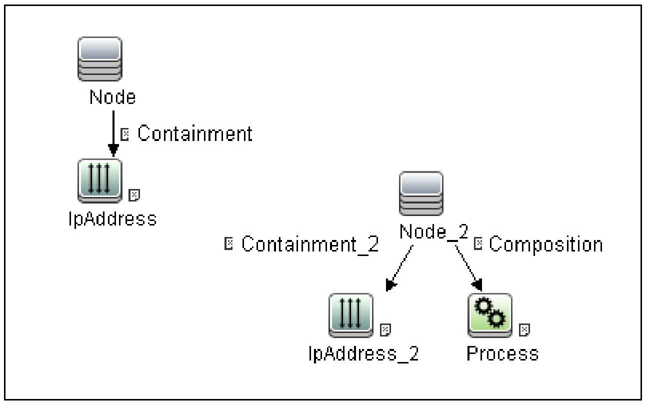
  The hostname of the ESX server is stored in the **name** (key) attribute of the **VMware ESX Server** CI.

- DFM clears all errors or warnings and returns all discovered results.

  Otherwise, if the connection is unsuccessful, DFM tries the next VIM protocol credential, until all are tried.

Trigger Query

- Trigger CI: Node
- Trigger query:



Adapter

This job uses the **VMware_ESX_Connection_by_VIM** adapter.

- **Adapter parameters**. The job uses two internal parameters - **runInSeparateProcess** and **remoteJVMClasspath** - whose values should remain unchanged.

Discovered CITs

- Composition
- VMware ESX Server

- Virtualization Layer Software

Troubleshooting and Limitations

- **Problem.** The following error message is displayed when an operation cannot be performed due to lack of permissions:

  ```
  User does not have required '<permission>' permission
  ```

  Solution. Check that permissions are set as **System.Read**.

- **Problem.** The following error message is displayed when credentials are not correct:

  ```
  Invalid user name or password
  ```

- **Problem.** The job completes with a time-out warning message:

  ```
  <<Progress message, Severity: Error>>
  VMware VIM: Timeout trying to connect to remote agent, try increasing credential
  timeout value
  ```

  Limitation. You cannot set the connection timeout value for the job, due to VMware API limitations. The default 60 seconds timeout is always used.

VMware ESX Topology by CIM Job

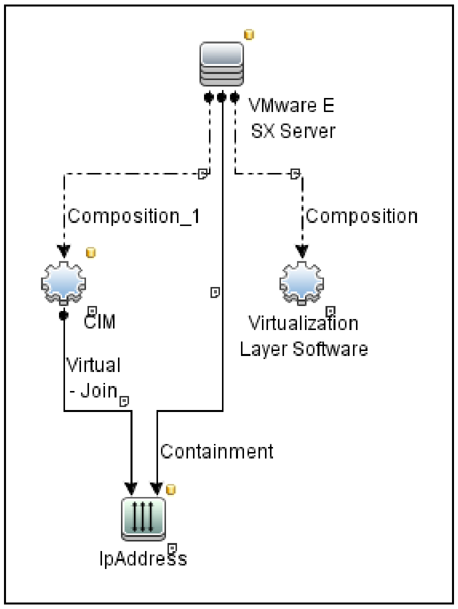
This job discovers the server inventory and virtualization topology using the CIM protocol.

This section includes:

- "Trigger Query" below
- "Adapter" on the next page
- "Discovered CITs" on page 250
- "Limitation" on page 250

Trigger Query

- Trigger CI: VMware ESX Server
- Trigger Query:



Node Name	Condition
CIM	NOT Reference to the credentials dictionary entry Is null AND CimCategory Contains "VMware"
VMware ESX Server	NOT BiosUuid Is null
IpAddress	NOT IP Probe Name Is null

Adapter

This job uses the **VMware_ESX_Topology_by_CIM** adapter.

- **Triggered CI Data**

Name	Value
credentialsId	${CIM.credentials_id}
esx_bios_uuid	${SOURCE.bios_uuid}
esx_cmdb_id	${SOURCE.root_id}
hypervisor_cmdb_id	${HYPERVISOR.root_id}
ip_address	${IP.name}

- **Adapter parameters:** none
- **Used Scripts:**
  - memory.py
  - host_discoverer.py
  - cim.py
  - cim_discover.py
  - vmware_cim_discover.py
  - vmware_cim_report.py
  - vmware_esx_topology_by_cim.py

Discovered CITs

- Composition
- Containment
- Cpu
- IpAddress
- Virtualization Layer Software
- VMware ESX Server
- Node
- VMware Host Resource
- ExecutionEnvironment

Limitation

Only BIOS UUID, Primary IP Address and hostname values may be available in virtualization namespace for virtual machines. If none of these values are present, VMs cannot be reported and are skipped.

VMware ESX Topology by VIM Job

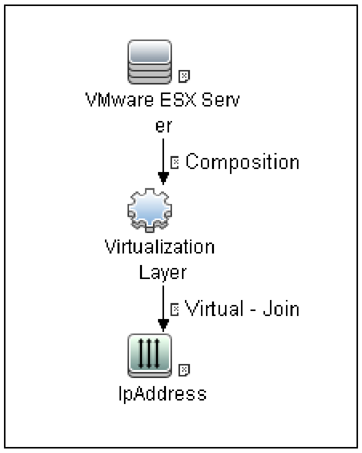
This job connects to ESX servers and discovers their topology using the VIM protocol.

This section includes:

- "VMware ESX Topology by VIM Job" above
- "VMware ESX Topology by VIM Job" above
- "VMware ESX Topology by VIM Job" above
- "VMware ESX Topology by VIM Job" above

Trigger Query

- Trigger CI: Virtualization Layer Software
- Trigger query:



Adapter

This job uses the **VMware_ESX_Topology_by_VIM** adapter.

- Triggered CI data:

credentialsId	The credentials ID of the VMware Infrastructure (VIM) protocol, saved in the ESX server attribute.
server_url	The URL for connection, taken from the ESX server **connection_url** attribute.
ip_address	The IP address of the ESX server.

- Adapter parameters:

Property	**Description**
remoteJVMClasspath	The class path used by the external java process. **Note:** Do not change this property from the default.
reportBasicTopology	Determines whether to report basic topology (true) or full topology (false). **Default:** false.
reportPoweredOffVMs	Determines whether to report powered off virtual machines (true) or not (false). **Default:** false.
runInSeparateProcess	When **true**, this enables the execution of the job in the external java virtual machine. **Default:** true. **Note:** Do not change this property from the default.

Discovered CITs

- Composition
- Containment
- Cpu

- Dependency
- ExecutionEnvironment
- FileSystemExport
- Interface
- IpAddress
- License Feature
- License Reservation
- License Server
- Node
- Usage
- VMware Datastore
- VMware ESX Server
- VMware Host Resource
- VMware Networking Policy
- VMware Port Group
- VMware Resource Pool
- VMware Virtual Switch
- Virtualization Layer Software

Troubleshooting

- **Problem.** The following error message is displayed when an operation cannot be performed due to lack of permissions:

  ```
  User does not have required '<permission>' permission
  ```

Check that permissions are set as **System.Read**.

- **Problem.** The following error message is displayed when credentials are not correct:

  ```
  Invalid user name or password
  ```

- **Problem.** The following warning message is displayed when DFM cannot retrieve licensing information due to insufficient permissions:

  ```
  User does not have required '<permission>' permission, licensing information
  won't be reported
  ```

VMware vCenter Connection by VIM Job

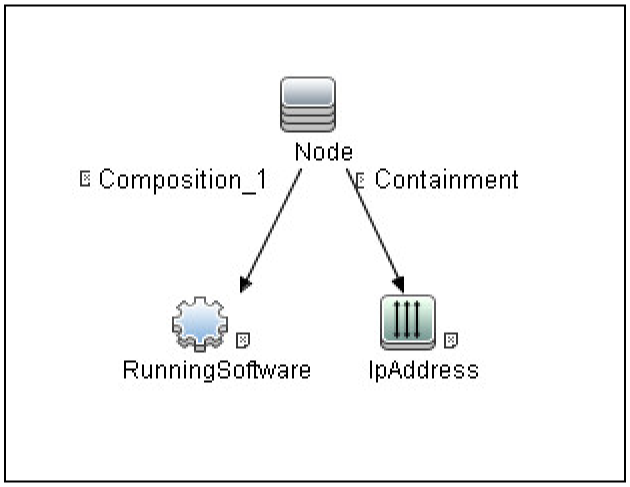
This job discovers vCenter or VirtualCenter Servers.

This section includes:

- "VMware vCenter Connection by VIM Job" above
- "VMware vCenter Connection by VIM Job" above
- "VMware vCenter Connection by VIM Job" above
- "VMware vCenter Connection by VIM Job" above

Trigger Query

- Trigger CI: Node
- Trigger query:



Adapter

This job uses the **VMware_VirtualCenter_Connection_by_VIM** adapter.

- Triggered CI Data:

Name	Description
connection_url	The connection URL of the vCenter server.
credentialsId	The credentials ID of the WMI agent CI.
ip_addresses	List of all IPs connected to Host.
rs_id	The CMDB ID of the vCenter server, reported as RunningSoftware.
vc_id	The CMDB ID of the vCenter server.

- Adapter Parameters

Property	Description
remoteJVMClasspath	The class path used by the external java process. **Note:** Do not change this property from the default.
runInSeparateProcess	When **true**, this enables the execution of the job in the external java virtual machine. **Default:** true. **Note:** Do not change this property from the default.

Discovered CITs

- Composition
- Containment
- IpAddress
- Node
- VMware VirtualCenter

Troubleshooting

- **Problem.** The following error message is displayed when an operation cannot be performed due to lack of permissions:

  ```
  User does not have required '<permission>' permission
  ```

  Solution. Check that the user has permissions for all entities being discovered: In the **VMware Infrastructure Client**, access the **Permissions** tab of each entity (host, cluster, VM, and so on). Verify that the user has been assigned at least a Read-Only role.

  > **Note:** You can view necessary permissions in the **Discovery Job Details** pane (**Universal Discovery > Discovery Modules/Jobs tab >** select **<job>** > **Details** tab).

- **Problem.** The following error message is displayed when credentials are not correct:

  ```
  Invalid user name or password
  ```

VMware vCenter Topology by VIM Job

This job connects to vCenter or VirtualCenter Servers and discovers the full VMware Infrastructure topology.

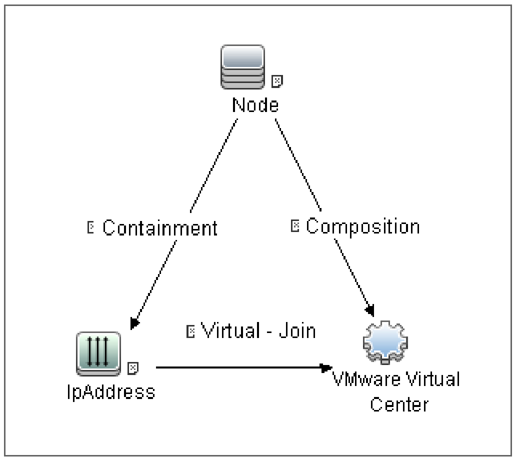
This section includes:

- "VMware vCenter Topology by VIM Job" above
- "VMware vCenter Topology by VIM Job" above
- "VMware vCenter Topology by VIM Job" above
- "Troubleshooting" on page 261

Trigger Query

- Trigger CI. VMware VirtualCenter.

- Trigger TQL query:



Adapter

This job uses the **VMware_VirtualCenter_Topology_by_VIM** adapter.

- Triggered CI Data:

Name	**Description**
credentialsId	The credentials ID of the VMware Infrastructure Management (VIM) protocol saved in the vCenter or VirtualCenter Server's attribute.
connection_url	The URL for connecting to VMware Infrastructure, taken from the vCenter or VirtualCenter Server's **connection_url** attribute.

- Adapter Parameters

Property	**Description**
remoteJVMClasspath	The class path used by the external java process. **Note:** Do not change this property from the default.

reportBasicTopology	Determines whether to report basic topology (true) or full topology (false). **Default:** false.
reportPoweredOffVMs	Determines whether to report powered off virtual machines (true) or not (false). **Default:** false.
runInSeparateProcess	When **true**, this enables the execution of the job in the external java virtual machine. **Default:** true. **Note:** Do not change this property from the default.

Discovered CITs

- Composition
- Containment
- Cpu
- Datacenter
- Dependency
- ExecutionEnvironment
- FileSystemExport
- Interface
- IpAddress
- Licence Feature
- License Reservation

- License Server
- Logical Volume
- Manage
- Membership
- Node
- Usage
- VMware Cluster
- VMware DAS Config
- VMware DPM Config
- VMware DRS Config
- VMware Datastore
- VMware Distributed Virtual Switch
- VMware ESX Server
- VMware Host Resource
- VMware Networking Policy
- VMware Port Group
- VMware Resource Pool
- VMware Uplink
- VMware Virtual Switch
- VMware Virtual Center
- Virtualization Layer Software

Troubleshooting

- **Problem:** The following error message is displayed when an operation cannot be performed due to lack of permissions:

  ```
  User does not have required '<permission>' permission
  ```

  Solution: Check that permissions are set as **System.Read**.

- **Problem:** The following error message is displayed when credentials are not correct:

  ```
  Invalid user name or password
  ```

- **Problem:** The following warning message is displayed in the Communication log during discovery:

  ```
  VM '<name>': powered off, VM is skipped
  ```

  Solution: This message indicates that the discovery found a powered-off VM. By default, powered-off VMs are not reported, mainly because the configuration of such powered-off VMs may be outdated. This outdated information can impact the identification of the VMs, so the topology reported might be incorrect.

  For example:

  - The MAC address of one of the interfaces might now be assigned to different VMs, yet still be listed for the powered-off VM.

  - The IP address might still be listed for the powered-off VM, but was reassigned to different machine by the DHCP server before discovery began.

  If you still want powered-off VMs to be reported, set the topology job's **reportPoweredOffVMs** parameter to **true**.

- **Problem:** The following warning message is displayed in the Communication log during discovery:

  ```
  Host '<name>': cannot find UUID, Host is skipped
  ```

  Solution: The UUID of the ESX server is a key attribute for the ESX server CI. It is not possible to report ESX server without a valid UUID. A UUID of the ESX server that consists of all zeros is also considered invalid. The message in the Communication log indicates that the specified ESX server was discovered but was skipped due to a missing or invalid UUID.

- **Problem:** The following warning message is displayed in the Communication log during discovery:

  ```
  VM '<name>': duplicate host key '<key>' found in another VM '<name>' which was
  preferred, VM is skipped
  ```

Solution: After all VMs are discovered, VMs containing duplicated host keys are filtered out. **host_key** is a key attribute of the VM, so it is not possible to report two VMs with the same host keys. The message in the Communication log indicates that there were duplicates found and one of the duplicated VMs was skipped.

If the **reportPoweredOffVMs** parameter is set to **true**, if the two VMs have different power statuses, the powered-on VM is preferred over the powered-off VM.

Chapter 17: VMware VMotion Discovery and Event Tracking

This chapter includes:

Overview

VMware VMotion technology moves an entire running VM instantaneously from one server to another. The VMware vCenter Server exposes a management interface that can be used by DFM to:

- Connect to vCenter Server using the VIM protocol, to discover its topology (Datacenters, Clusters, ESX Servers, Resource Pools, Virtual Machines, and so on).
- Connect to ESX Server and discover its full topology. This discovery is limited to the server itself.
- Listen for events that occur in the inventory structure. Currently two types of events are tracked and reported:
  - VMotion events, when the VM migrates from server to server.
  - VM powering-on event, when the VM is turned on.

VMware provides an SDK describing this interface, which includes documentation, API reference, libraries, and examples. VMware Infrastructure SDK can be downloaded from http://www.vmware.com/support/developer/vc-sdk/.

Supported Versions

This discovery supports:

- VirtualCenter 2.5, 2.0, vCenter Server 4, 4.1, 5.0, and 5.1
- ESX Server 3.0, 3.5, 4.0, 4.1, 5.0, and 5.1

How to Discover VMware VMotion and Track Events

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   To connect to any server using the VIM protocol, prepare the following:

- A connection URL, for example, **https://vcserver/sdk**.
- Credentials (user name and password). A user account must be created for you on the VMware server.

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Set up permissions**

   VMotion event-driven discovery requires special permissions for the protocol used:

   - **System.Read** permissions for the user performing the login, for all DFM actions. The user must be a member of the **Read-Only** user group.

3. **Run the discovery**

   For details on running jobs, see "Module/Jobe-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   a. Run the **VMware vCenter Connection by VIM** and **VMware vCenter Topology by VIM** jobs.

   b. Run the **VMware vMotion Monitor by VIM** job. The job includes the **VMware_VMotion_discovery_by_VIM** adapter that listens for VM migration events collected by the VirtualCenter server.

VMware vMotion Monitor by VIM Job

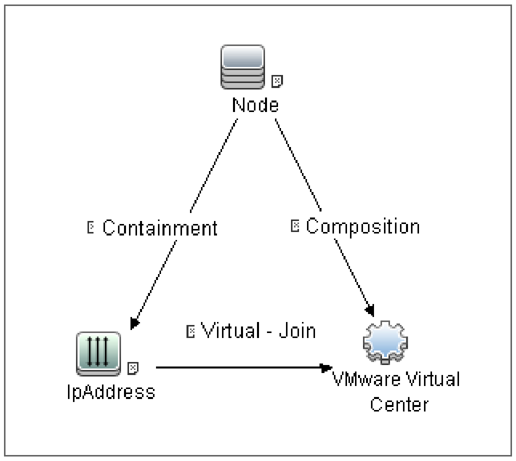
This section includes:

- "VMware vMotion Monitor by VIM Job" above
- "VMware vMotion Monitor by VIM Job" above
- "VMware vMotion Monitor by VIM Job" above

Trigger Query

- Trigger CI: VMware VirtualCenter

- Trigger query:



Adapter

This job uses the **VMware_VMotion_discovery_by_VIM** adapter.

- Triggered CI Data:

Name	Value	Description
credentialsId	${SOURCE.credentials_id}	The credentials ID of the VIM protocol saved in the VirtualCenter attribute.
ip_address	${SOURCE.application_ip}	The IP address, taken from the vCenter Server's **application_ip**.
server_url	${SOURCE.connection_url}	The URL for connection, taken from the vCenter Server's **connection_url** attribute.

- Adapter Parameters:

Property	Description
connectionRetryNumber	The maximum number of times that DFM attempts to restore the connection. **Default:** 0 (zero), meaning the number of attempts is unlimited.

Property	Description
eventBasedDiscoveryEnabled	If this parameter is set to **true**, every time the job is activated, it stays connected to the destination machine listening for VMotion events, until the job is stopped. **Default:** true.
historyHours	The period within which DFM checks for untracked VMotion events. DFM calculates the period from when the job is activated going backwards in time. **Default: 24 hours**.
remoteJVMClasspath	The class path used by the external java process. **Note:** Do not change this property from the default.
runInSeparateProcess	When **true**, this enables the execution of the job in the external java virtual machine. **Default:** true. **Note:** Do not change this property from the default.

Discovered CITs

- Composition
- Containment
- ExecutionEnvironment
- Interface
- IpAddress
- Node

- Usage
- VMware Host Resource
- VMware Port Group
- VMware Virtual Switch
- Virtualization Layer Software

Chapter 18: VMware Discovery Troubleshooting and Limitations

This chapter includes:

Troubleshooting

- **Problem.** The following error message is displayed:

  ```
  Required class %s not found. Verify VMware SDK jar files are present in probe.
  See documentation for details.
  ```

  Cause. The SDK *.jar files are not copied to the Data Flow Probe.

  Solution. Copy the *.jar files as described in "VMware Prerequisites" on page 223.

- **Problem.** The following error message is displayed:

  ```
  User does not have required 'System.Read' permission
  ```

  Cause. There is a lack of permissions from the user account when DFM connects to the ESX server's vCenter Server.

  Solution.

  a. Verify that credentials are defined for the VMware Infrastructure Management (VIM) protocol in the proper priority, so that credentials with full permissions have a lower index number than credentials with less permissions. For details, see "Index" in the HP Universal CMDB Data Flow Management Guide.

  b. If DFM previously discovered connections using credentials with less than full permissions, you must rerun the connection job.

     - For ESX connection and topology: run **VMware ESX Connection by VIM**) to update the `credentials ID attribute` of ESX server, and then run the topology job **VMware ESX Topology by VIM)**.

     - For vCenter topology: edit the integration point and choose credentials with more permissions.

Limitations

- DFM can discover the total number of licenses and available licenses for each feature, but only when the user has **Global.Licenses** permission. If the user does not have such permissions, these attributes of the `License Feature` CI are not populated.

- Different versions of ESX Servers (versions 3.0 and 3.5) report the `feature_is_edition` flag differently for the `esxFull` feature: for the older version it is reported as `false` and for the newer version it is reported as `true`. Because of this discrepancy, DFM does not report this attribute.
- Different versions of ESX Servers (versions 3.0 and 3.5) report the total or available license counts differently for ESX-specific features (`nas`, `iscsi`, `vsmp`, `san`) that are included in the `esxFull` edition license. For these features, DFM does not report these attributes.
- There is a difference between the VMware protocol versions: certain attributes appear only in newer versions and do not appear in previous versions. As a result, when using an old protocol certain attributes are not discovered, especially for clusters and licenses.
- DFM does not discover or report licensing information for vCenter\ESX server version 4.0 or above.
- DFM does not report information about the order of teamed interfaces. You can group server physical interfaces of an ESX server into NIC Teaming groups, while specifying the order of such interfaces in a group (first, second, and so on). Information about what interface are teamed is reported but the order of these interfaces is not.

Chapter 19: Xen and KVM Discovery

This chapter includes:

Overview

Xen is a Hypervisor providing services that allow multiple computer operating systems to execute concurrently on the same computer hardware. Xen is currently available for the IA-32, x86-64 and ARM computer architectures.

Kernel-based Virtual Machine (KVM) is a virtualization infrastructure for the Linux kernel. KVM supports native virtualization on processors with hardware virtualization extensions. KVM has also been ported to FreeBSD and Illumos in the form of loadable kernel modules.

KVM originally supported x86 and x86-64 processors and has been ported to S/390, PowerPC, and IA-64. An ARM port is in progress, KVM hypervisor porting to ARM Cortex-A15 is made available by Virtual Open Systems.

libvirt is an open source API, daemon and management tool for managing platform virtualization. You can use it to manage Linux KVM, Xen, VMware ESX and other virtualization technologies. Graphical interfaces use it, such as Virtual Machine Manager, as do command line interfaces (virsh), and higher level tools like oVirt.

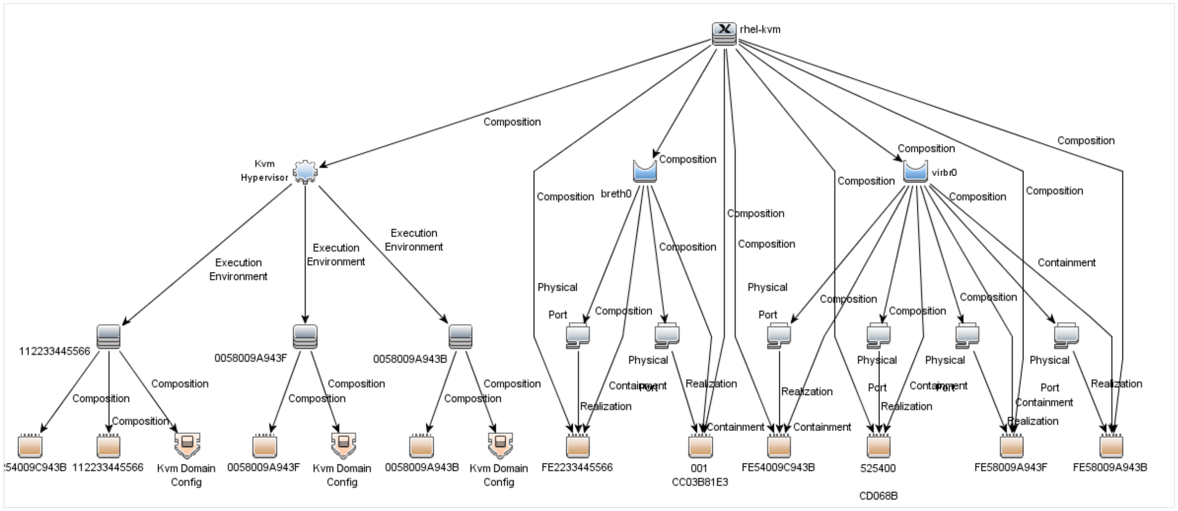
This package discovers Xen and KVM virtualization solutions using libvirt as an API via shell connection.

Supported Versions

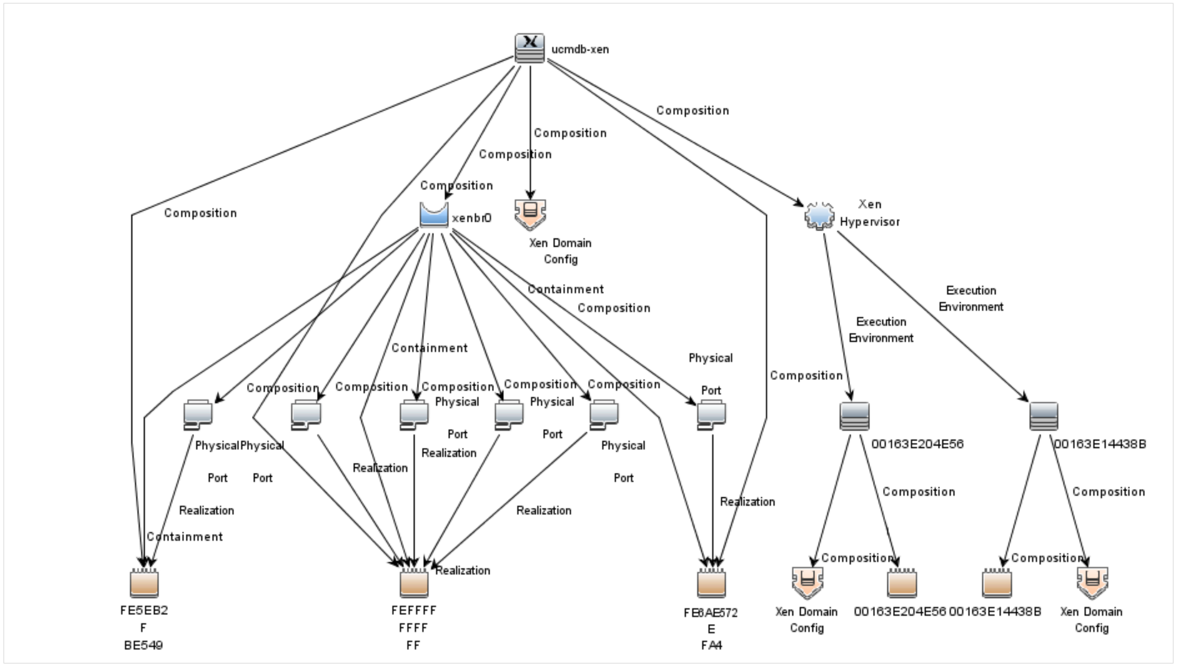
libvirt provides a unified API to manage Xen and KVM. This discovery supports libvirt version 0.8.2 to 0.8.x and version 0.9.1 to 0.9.6.

Topology

The following image displays the topology of KVM discovery:



The following image displays the topology of Xen discovery:



Discovery Mechanism

A regular connection by shell is performed to the destination machine running Xen or KVM with installed libvirt managing tools. Using the libvirt CLI, you can discover details about the destination machine.

How to Discover KVM and Xen

This discovery is performed in the following stages:

- Discover generic Linux hosts
- Discover virtualization for XEN and KVM

1. **Prerequisite - Set up protocol credentials**

   You must set up the SSH or Telnet protocol. For either protocol, you must prepare a user name and password.

2. **Run the discovery**

   a. Run the **Range IPs by ICMP** job to discover IP CIs.

   b. Run the **Host Connections by Shell** job to discover the target host and shell connectivity to it.

   c. Run the **Xen and KVM by Shell** job to discover topology for Xen and KVM.

Adapter

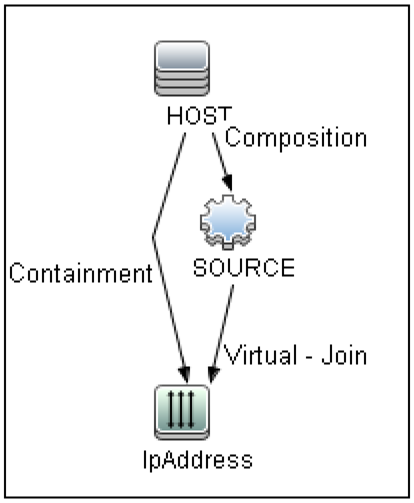
This discovery uses the Xen and KVM by Shell adapter.

Input CIT

Shell

Input TQL Query

The following graphic shows an input TQL query for this job.



Attribute	Condition
HOST	NOT CI Type Equal nt
IpAddress	NOT IP Probe Name Is null

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
credentialsId	${SOURCE.credentials_id}
hostId	${SOURCE.root_container}
ip_address	${SOURCE.application_ip}

Used Script

xen_by_tty.py

Discovered CITs

- Bridge
- Composition
- Containment
- ExecutionEnvironment
- FileSystem
- FileSystemExport
- Interface
- Kvm domain config
- Layer2Connection
- Node
- PhysicalPort
- Realization
- Virtualization Layer Software
- Xen domain config

Parameters

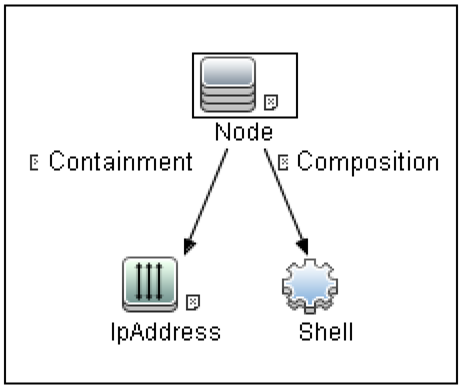
Parameter	Type	Description
virsh_path	String	The full path to the virsh management utility.

Xen and KVM by Shell Job

Adapter

This job uses the Xen and KVM by Shell adapter.

Trigger Query



Parameters

Parameter	Type	Description
virsh_path	String	The full path to the virsh management utility.

Discovery Flow

1. **Discover list of domains**

   virsh list sample output:

```
Id Name                 State
----------------------------------
0 Domain-0             running
```

```
15 ucmdb-vm-vista       idle
16 ucmdb-vm-xp          idle
```

2. **Discover list of domain configurations**

   The following command creates an xml file with the configuration:

   virsh dumpxml <domain_name>

   a. Distinguishing hypervisor:

      Xen : <domain type='xen' id='15'>

      KVM: <domain type='kvm' id='12'>

   b. VM memory configuration

      Xen and KVM : <memory>1572864</memory> <currentMemory>1572864</currentMemory>

   c. VM CPU configuration

      Xen and KVM: <vcpu>1</vcpu>

   d. VM Networking configuration

      Xen and KVM: <interface type='bridge'><mac address='00:16:3e:20:4e:56'/><source bridge='xenbr0'/><script path='vif-bridge'/><target dev='vif15.0'/></interface>

Part 4: Clustering and Load Balancing > Failover Clusters

Chapter 20: EMC AutoStart Discovery

This chapter includes:

Overview

EMC AutoStart provides high availability for multiple operating systems to deal with service outages - planned or unplanned.

The EMC AutoStart discovery process allows you to discover a full topology.

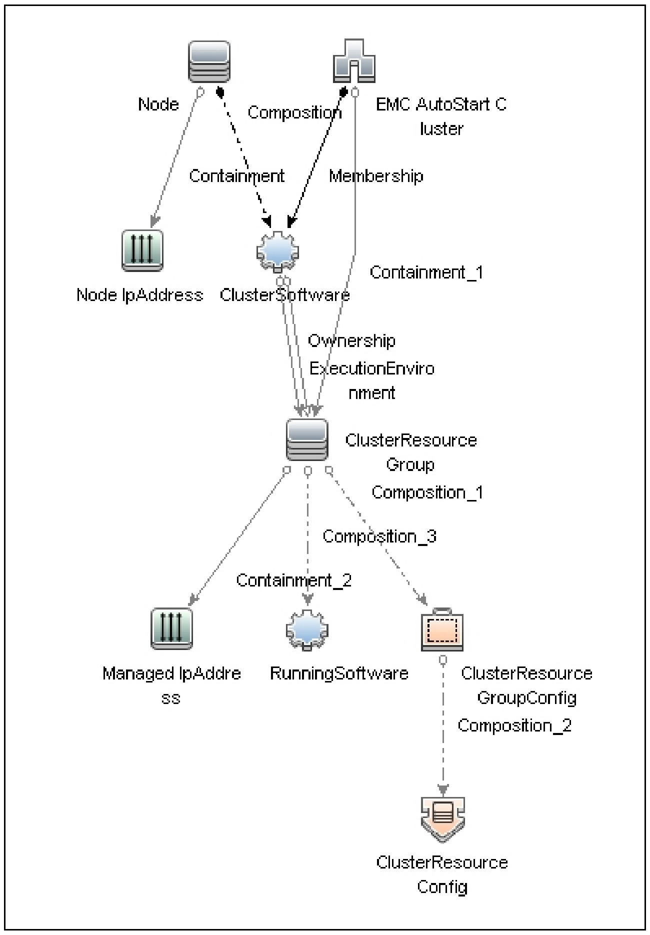
Supported Versions

EMC AutoStart discovery supports version 5.x of EMC AutoStart.

Topology

The following image displays EMC AutoStart topology.

Note: For a list of discovered CITs, see "Discovered CITs" on page 287.


Node
Composition
EMC AutoStart Cluster
Containment
Membership
Node IpAddress
ClusterSoftware
Containment_1
Ownership
ExecutionEnvironment
ClusterResourceGroup
Composition_1
Composition_3
Containment_2
Managed IpAddress
RunningSoftware
ClusterResourceGroupConfig
Composition_2
ClusterResourceConfig

How to Discover EMC AutoStart

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the following protocols:

   - **SSH**
   - **Telnet**
   - **NTCMD**

     For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   a. Ensure there is Shell connectivity to one or more nodes of the AutoStart domain.

   b. If required, configure sudo on each target host to allow execution of all commands used. See "EMC AutoStart Discovery Commands" on page 291.

   c. In Windows, if connecting with **NTCMD**, run the **HPCmdSvc** service as a user recognized by AutoStart. Otherwise, configuration information is unavailable.

3. **Run the Discovery**

   a. Run the **Range IPs by ICMP** job in order to discover the target IPs.

   b. Run the **Host Connection by Shell** job in order to discover the target host and shell connectivity to it.

   c. Run the **Host Applications by Shell** job in order to discover applications of the target host, including EMC AutoStart Cluster software and agent processes.

   d. Run the **EMC AutoStart by Shell** job in order to discover the topology of the target EMC AutoStart cluster.

EMC AutoStart by Shell Job

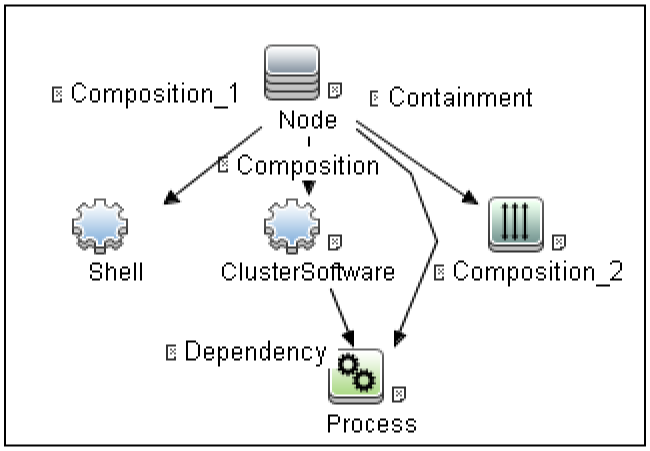
This section gives details about the EMC AutoStart by Shell job.

Adapter

This job uses the **EMC_AutoStart_by_Shell** adapter.

Trigger Query

emc_autostart_with_shell



Node Name	Condition
Shell	NOT Reference to the credentials dictionary entry Is null AND NOT Application IP Is null
IpAddress	NOT IP Probe Name Is null
Process	NOT Process Path Is null AND Name Like "ftAgent%"
ClusterSoftware	DiscoveredProductName Equal "EMC AutoStart Cluster SW" AND NOT Name Is null
Node	None

Parameters

This job uses parameter values from the adapter. By default, parameters are not overridden.

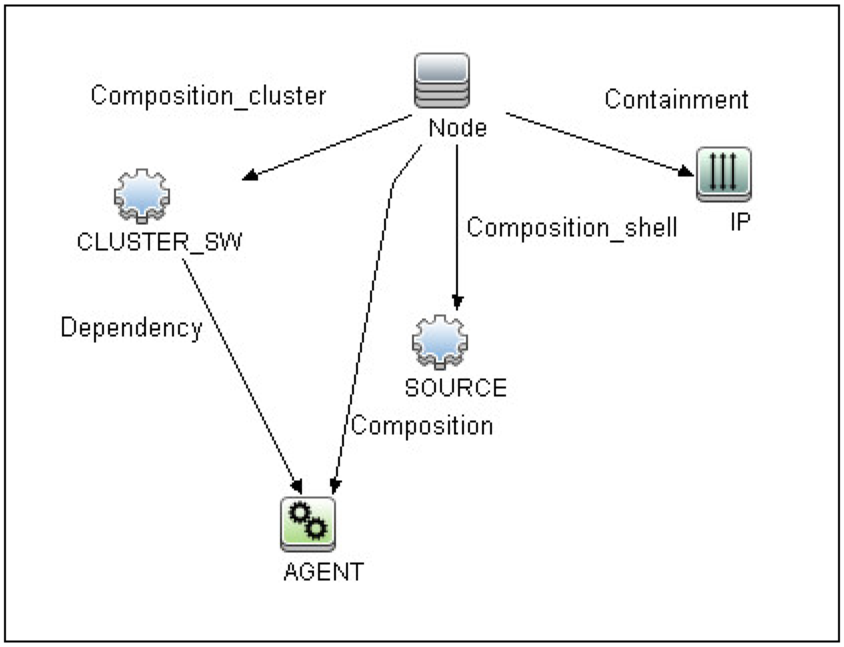
EMC_AutoStart_by_Shell Adapter

This section gives details about the **EMC_AutoStart_by_Shell** adapter.

Input CIT

Shell

Input TQL Query



Node Name	Condition
SOURCE	NOT Reference to the credentials dictionary entry Is null AND NOT Application IP Is null
IP	NOT IP Probe Name Is null
AGENT	NOT Process Path Is null AND Name Like "ftAgent%"
CLUSTER_SW	DiscoveredProductName Equal "EMC AutoStart Cluster SW" AND NOT Name Is null
Node	None

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
credentialsId	${SOURCE.credentials_id}
agentPath	${AGENT.process_path}
ip_address	${SOURCE.application_ip}
domainName	${CLUSTER_SW.name}

Used Scripts

- emc_autostart.py
- emc_autostart_discover.py
- emc_autostart_report.py
- emc_autostart_by_shell.py

Discovered CITs

- ClusterResourceConfig
- ClusterResourceGroup
- ClusterResourceGroupConfig
- ClusterSoftware
- Composition
- Containment
- EMC AutoStart Cluster
- ExecutionEnvironment
- IpAddress

- Membership
- Node
- Ownership

Discovery Flow

This section describes the discovery flow of the **EMC Autostart by Shell** job.

1. **Calculate paths**

   The path of the **ftAgent** process discovered by **Application Signatures** is analyzed. These paths are calculated:

   - root of deployment
   - path to folder with executable files (bin)

2. **Verify presence of ftcli command**

   Execute command **ftcl** with **-version** argument to:

   - Verify the command is available by calculated path.
   - Get version information about installed EMC AutoStart software.

3. **Verify domain name**

   Domain name calculated from command line of EMC AutoStart software processes should be verified.

   - The job tries to read the configuration file **<root>/config/<domain-name>-sites**.
   - If the file is missing, the domain name is considered invalid and the job ends.

4. **Discover cluster topology**

   The command **ftcli** is used to read configuration of the cluster, including:

   - **nodes** (listNodes, getNode)
   - **managed IPs** (listManagedIPS, getIP)

- **managed NICs** (listManagedNics, getNic)
- **resource Groups** (listResourceGroups, getResourceGroup)
- **data sources** (getDataSource)
- **processes** (getProc)

EMC AutoStart Discovery Commands

This section describes the commands used by EMC AutoStart Discovery.

Command ftcli.exe -version

```
"C:\Program Files\EMC\AutoStart\DDM_dom\bin\ftcli.exe" -version
```

Output

```
Version 5.4.1 Build 82

                                EMC AutoStart
                            Version 5.4.1 build 82
                        Built: Thu Nov 3 16:09:59 EDT 2011
```

Command ftcli.exe -cmd "listNodes"

```
"C:\Program Files\EMC\AutoStart\DDM_dom\bin\ftcli.exe" -cmd "listNodes"
```

Output

```
Node                      Type            State
-----------------------   ------------    --------------
ddm-autostart             Primary         Agent Running
ddm-autostart2            Primary         Agent Running
```

Command ftcli -cmd "getNode node1"

```
/opt/EMCas/bin/ftcli -cmd "getNode node1"
```

Output

```
Description       : Entry for node node1
System Name       : node1
Operating System  : HP-UX 11.31
Kernel Arch       : ia64
Main Memory (MB)  : 4076
Swap space  (MB)  : 24506
Supported DS      :
IP Address(es)    : 10.20.30.136
                    10.20.30.137
Node Attributes   : name=Ticket              value=1
LAAM Version      : 5.4.1
LAAM Version Info : Version 5.4.1 build 82
```

```
Build Date        : Thu Nov 3 16:09:30 EDT 2011
State             : Agent Running
```

Chapter 21: IBM High Availability Cluster Multiprocessing (HACMP) Discovery

This chapter includes:

Overview

High Availability Cluster Multiprocessing (HACMP) is an IBM solution for high-availability clusters on the AIX UNIX and Linux for IBM System p platforms.

HACMP can run on up to 32 computers or nodes, each of which is either actively running an application (active) or waiting to take over should another node fail (passive). Data on file systems can be shared between systems in the cluster.

HACMP relies heavily on IBM's Reliable Scalable Cluster Technology (RSCT). RSCT includes daemons that are responsible for monitoring the state of the cluster (for example, a node, NIC or network crash) and for coordinating the response to these events. HACMP is an RSCT aware client. RSCT is distributed with AIX.

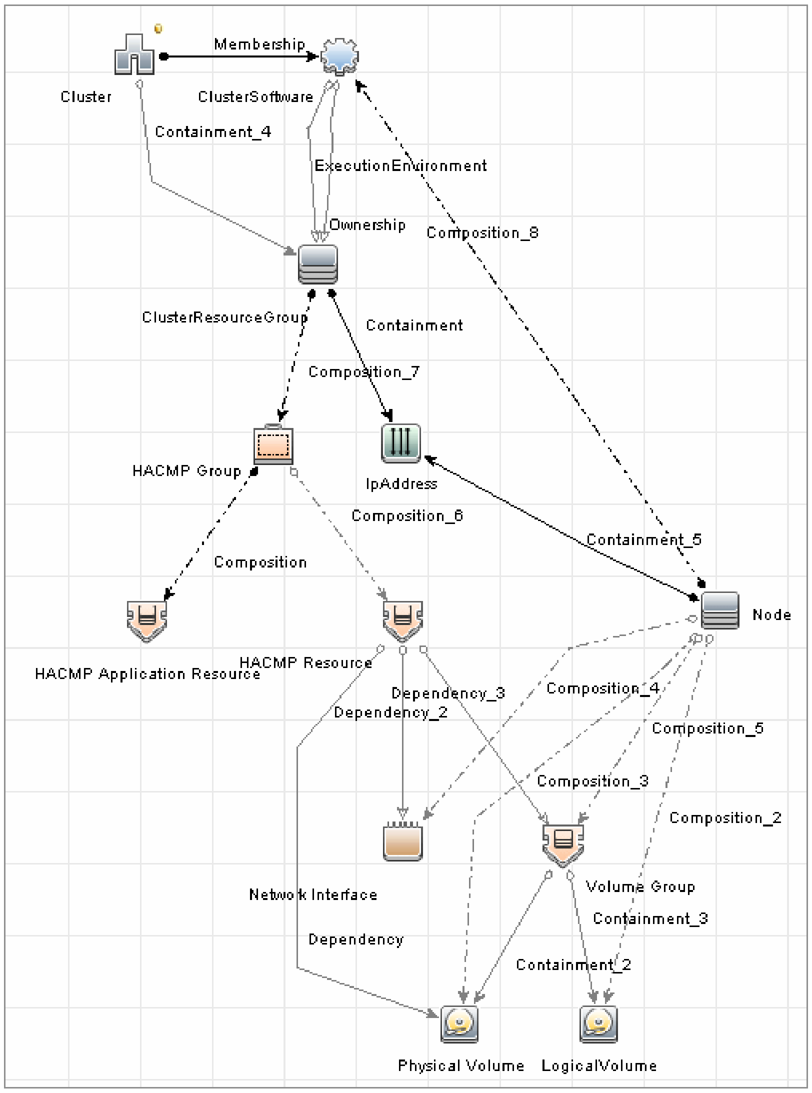
The **IBM_HACMP** package discovers HACMP on AIX via TTY (SSH or Telnet protocols). The package follows the discovery model to discover the HACMP Topology (configured networks, node interfaces-both public TCP/IP and serial heartbeat, and service IPs) and Application Resources (configured resource groups, application servers, and volume groups). The package maps the configured public interfaces to UCMDB IPs, serial interfaces to directories beneath the UCMDB hosts, as well as volume groups to logical disks beneath the UCMDB host, and Application Resources to the Topology.

Supported Version

This discovery supports HACMP 5.4 on AIX 5.3.

Topology

The following image displays the topology of the HACMP discovery.



How to Discover IBM HACMP

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the following Shell protocols:

   - SSH Protocol
   - Telnet Protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   - Verify that the Host Connection adapters have been successfully run on the nodes involved in the cluster.

     For details, see "Network - Basic Discovery" on page 925.

   - Load the Storage Topology add-on package prior to deployment of the HACMP package.

3. **Run the Discovery**

   a. Verify that the Probe has an IP range assigned to it that includes the IPs of the target machines running IBM HACMP Cluster.

   b. Verify that the Shell (SSH or Telnet) credentials are specified. For details, see "Prerequisite - Set up protocol credentials" above.

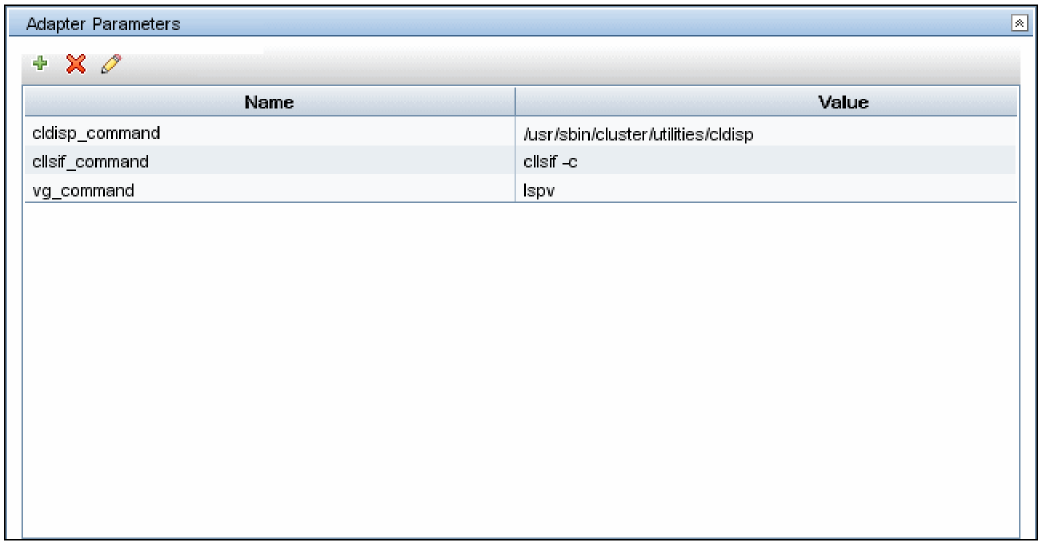
   c. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   d. Run the **Host Connection by Shell** job to discover Shell connectivity and basic information about the hosts.

   e. Verify that the **Host Connection** jobs have previously discovered the hosts that are to be part of the HACMP cluster. For details, see "Prerequisite - Set up protocol credentials" above. If you have not yet run these jobs, you can activate them now.

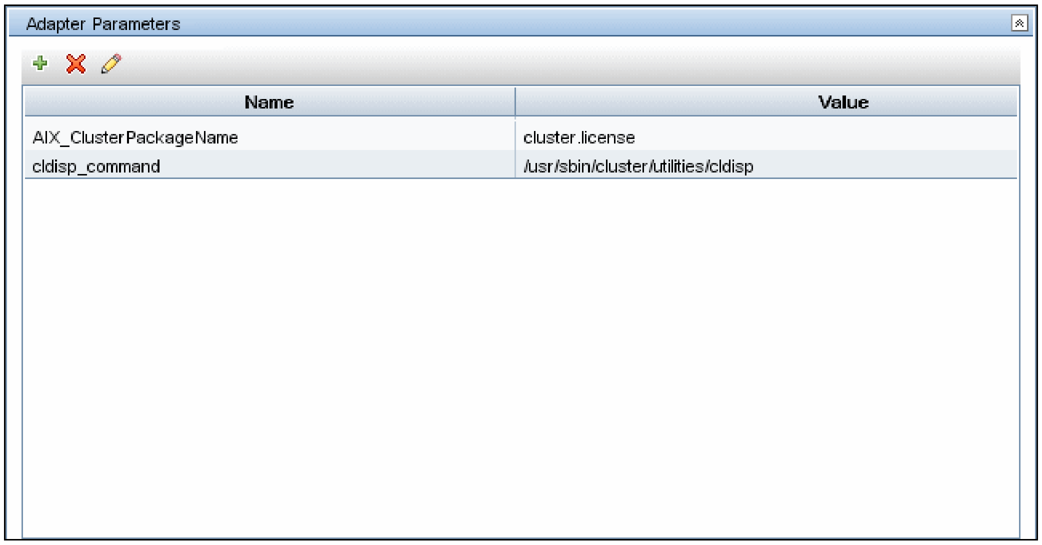
   f. Check the adapter parameters for the HACMP Topology and Application Discovery adapters. To use **sudo** with the commands, adjust the parameters appropriately. They can also be adjusted on the job.

HACMP Application discovery adapters



Name	Value
cldisp_command	/usr/sbin/cluster/utilities/cldisp
cllsif_command	cllsif -c
vg_command	lspv

HACMP Topology discovery adapters



Name	Value
AIX_ClusterPackageName	cluster.license
cldisp_command	/usr/sbin/cluster/utilities/cldisp

g. Activate the **HACMP Topology Discovery** job. After the job completes, verify the creation of **HACMP** CIs through the Discovery Results pane. For details on the CIs that are discovered, see the section describing discovery progress and results in the *HP Universal CMDB Data Flow Management Guide*.

h. Activate the **HACMP Application Discovery** job. This job creates HACMP application and resource CIs.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Discovery Mechanism

This section describes the following commands:

- "Verify that the Connected OS Supports HACMP" below
- "Get the Version of HACMP" below
- "Get Cluster Information" on the next page
- "Get DNS Information from the Host File" on the next page
- "Get Volume Group Information" on page 301
- "Get HACMP Application Information" on page 302

Verify that the Connected OS Supports HACMP

Command	uname
Example of output	`aix`
Values taken	aix
Comments	This command retrieves the OS. This package runs only on AIX platforms so Discovery must verify the OS.

Get the Version of HACMP

Command	lslpp -l cluster.license
Example of output	`cluster.license 5.4.0.0 COMMITTED HACMP Electronic License`
Values taken	5.4.x.x
Comments	This command gives the HACMP version. Discovery verifies that the HACMP version is valid.

Get Cluster Information

Command	/usr/sbin/cluster/utilities/cldisp

Example of output	`## =========================================`
	`## Cluster: db590_db591`
	`## Cluster services: active`
	`## State of cluster: up`
	`## Substate: stable`
	`##`
	`## #############`
	`## APPLICATIONS`
	`## #############`
	`## ...`
	`## =========================================`

Values taken	Cluster: db590_db591
Comments	This command retrieves the HACMP Cluster name.

Get DNS Information from the Host File

Command	cat /etc/hosts

Example of output	`## Sample output...`
	`## ========================================`
	`## # Do not remove the following line, or various programs`
	`## # that require network functionality will fail.`
	`## 127.0.0.1 testserver localhost.localdomain localhost`
	`## 12.20.30.3 server1 server1.compay.net`
	`## 12.20.20.3 server1-backup server1-backup.company.net`
	`## 192.168.1.103 server1-local server1-local.company.net`
	`## 12.20.30.4 server2 server1.compay.net`
	`## 12.20.20.4 server2-backup server2-backup.company.net`
	`## 192.168.1.104 server2-local server2-local.company.net`
	`## ========================================`

Values taken	IP Address and name
Comments	This command retrieves the host name and the IP.

Get Volume Group Information

Command	lspv

Example of output	`## Sample output...`
	`# dwscmdb : lspv`
	`# hdisk1 00ca4bbe84bdab4f rootvg active`
	`# hdisk0 00ca4bbe84bdac14 rootvg active`
	`# hdisk2 00ca4bbeeeb6b3c2 QSWIQ9A0_vg concurrent`
	`# hdisk3 00ca4bbeeeb3c581 None`
	`# hdisk4 00ca4bbeeeb6b499 QSWIQ9A0_vg concurrent`
	`# hdisk5 00ca4bbeeeb3c403 None`
	`# hdisk6 00ca4bbeeeb6b60d QSWIQ9B0_vg concurrent`
	`# hdisk7 00ca4bbeeeb3c4c2 QSWIQ9B0_vg concurrent`
	`# hdisk8 00ca4bbeeeb6b84f QSWIQ9A0_vg concurrent`
	`# hdisk9 00ca4bbeeeb6b920 QSWIQ9A0_vg concurrent`
	`# hdisk10 00ca4bbeeeb3c641 None`
	`# hdisk11 00ca4bbeeeb3c7c0 None`
	`# hdisk12 00ca4bbeeeb6b6e5 QSWIQ9B0_vg concurrent`
	`# hdisk13 00ca4bbeeeb3c700 QSWIQ9B0_vg concurrent`

Values taken	Volume group name
Comments	This command retrieves the volume groups.

Get HACMP Application Information

Command	cldisp

Example of output

```
## Sample output...
 ## ========================================
 ## Cluster: db590_db591
 ## Cluster services: active
 ## State of cluster: up
 ## Substate: stable
 ##
 ## #############
 ## APPLICATIONS
 ## #############
 ## Cluster sy008_sy015 provides the following applications:
assy008
 ## Application: assy008 {online}
 ## This application is part of resource group 'ressy008'.
 ## Resource group policies:
 ## Startup: on home node only
 ## Fallover: to next priority node in the list
 ## Fallback: never
 ## Nodes configured to provide assy008: a_wwasy008 {up} b_
ddasy015 {up}
 ## Node currently providing assy008: a_wwasy008 {up}
 ## The node that will provide assy008 if a_wwasy008 fails is: b_
ddasy015
 ## assy008 is started by /usr/local/bin/start_assy008
 ## assy008 is stopped by /usr/local/bin/stop_assy008
 ## Resources associated with assy008:
 ## Service Labels
 ## wwasy008(141.122.74.142) {online}
 ## Interfaces configured to provide wwasy008:
 ## wwasy008-boot {down}
 ## with IP address: 141.122.74.149
```

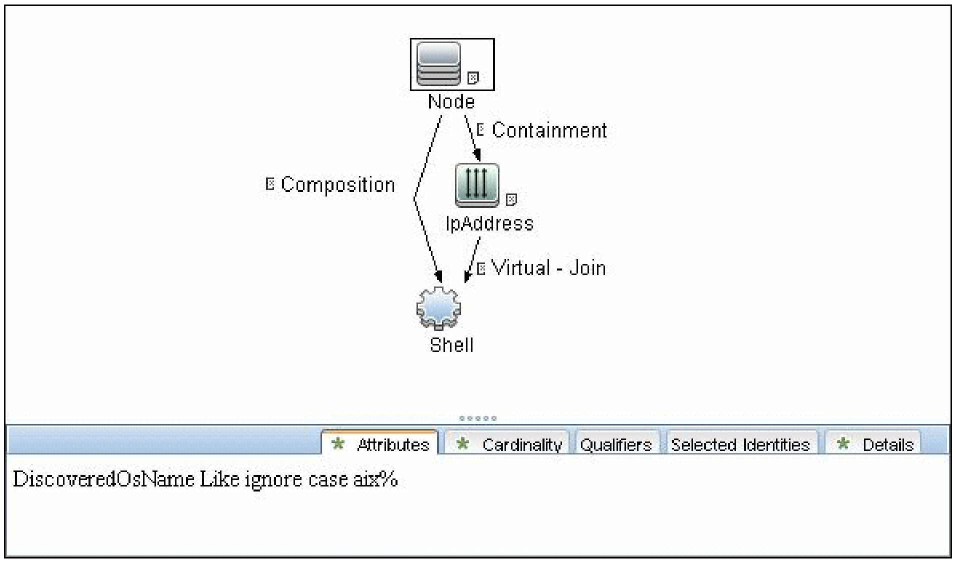
	`## on interface: en1`
	`## on node: a_wwasy008 {up}`
	`## on network: net_ether_01 {up}`
	`## wwasy008-stdby {up}`
	`## with IP address: 192.168.2.40`
	`## on interface: en2`
	`## on node: a_wwasy008 {up}`
	`## on network: net_ether_01 {up}`
	`## ddasy015 {up}`
	`## with IP address: 141.122.74.154`
	`## on interface: en1`
	`## on node: b_ddasy015 {up}`
	`## on network: net_ether_01 {up}`
	`## ddasy015-stdby {up}`
	`## with IP address: 192.168.2.10`
	`## on interface: en2`
	`## on node: b_ddasy015 {up}`
	`## on network: net_ether_01 {up}`
	`## Shared Volume Groups:`
	`## vg100`
	`## vg199`
	`## No application monitors are configured for assy008.`
	`##`
	`## #############`
	`## TOPOLOGY`
	`## #############`
	`## ...`
	`## ================`

Values taken	Application information
Comments	This command retrieves the HACMP Application information.

HACMP Topology Discovery Job

Trigger Query (Shell not NTCMD HACMP)

This trigger requires a TTY Shell that is not an NTCMD Shell.



DiscoveredOsName Like ignore case aix%

Adapter

Created/Changed Entities

- Hacmpcluster CIT
- Failoverclustersoftware CIT
- Logical Volume
- Physical Volume
- Volume Group
- Network Interface

Used Scripts

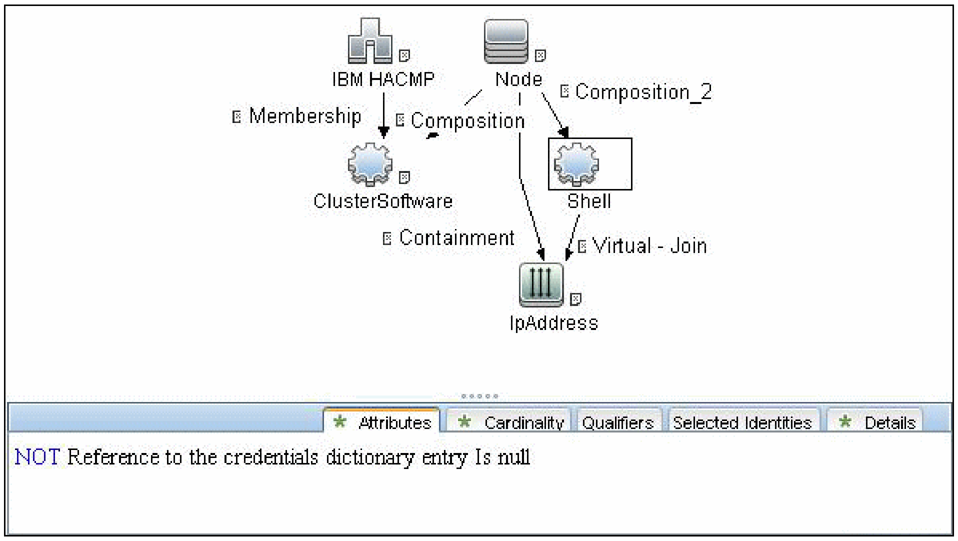
- storage_topology.py
- TTY_HACMP_Topology.py

Discovered CITs

- ClusterSoftware
- Composition
- Containment
- HACMP Cluster
- Interface
- IpAddress
- LogicalVolume
- Membership
- Node
- Physical Volume
- Volume Group

HACMP Application Discovery Job

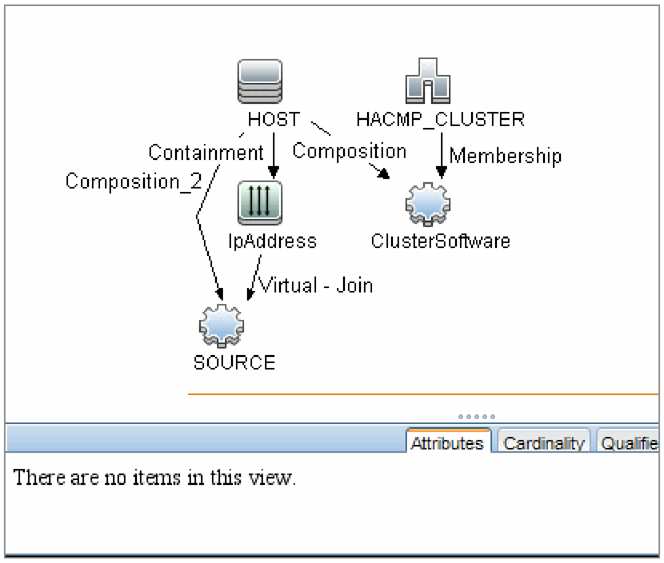
Trigger Query (Shell in HACMP Cluster)



NOT Reference to the credentials dictionary entry Is null

Adapter

- **Input Query**



There are no items in this view.

- **Created/Changed Entities**

- Hacmpgroup
- Hacmpresource
- Network Interface
- Cluster Server
- IpAddress
- Physical Disk
- Volume Group

Used Script

- TTY_HACMP_Applications.py

Discovered CITs

- ClusterResourceGroup
- ClusterSoftware
- Composition
- Containment
- Dependency
- ExecutionEnvironment
- HACMP Cluster
- HACMP Resource
- HACMP Resource Group
- Interface
- IpAddress
- Membership

- Node
- Ownership
- Physical Volume
- RunningSoftware
- Usage
- Volume Group

Chapter 22: Microsoft Cluster Discovery

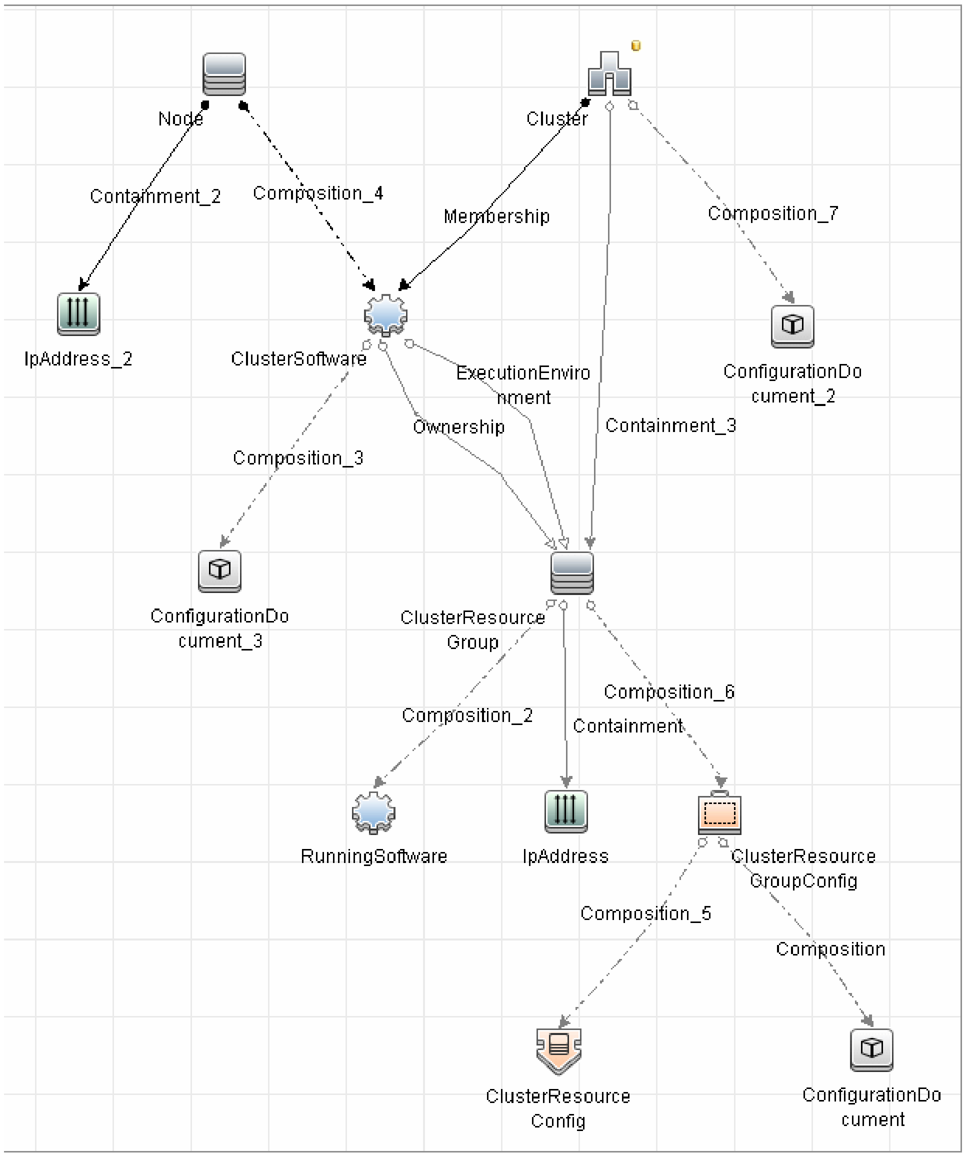
This chapter includes:

Microsoft Cluster Server View Topology

The Microsoft Cluster Server View shows the MS Cluster and the cluster software (the agents running on the actual host) as its members.

The cluster is composed of several `Clustered Servers` that are the virtual hosts or servers providing the platform for the virtual service used by the cluster clients (through the virtual IPs). The cluster contains Microsoft Cluster Groups. Each of the groups contains Microsoft Cluster Resources. For each Cluster Resource Group, it is assumed that different, dedicated, virtual IPs are being assigned; these IPs are configured for the use of the cluster clients.

Note: For a list of discovered CITs, see "Discovered CITs" on page 315.



Supported Versions

- Windows Server 2003
- Windows Server 2008

How to Discover Microsoft Cluster Servers

The MS Cluster discovery process enables you to discover the topology of a Microsoft Cluster Server on the network.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the WMI and NTCMD or PowerShell protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   Activate the relevant jobs in the following order:

   a. **Host Connection by Shell** or **Host Connection by PowerShell**

   b. **Host Applications by Shell/SNMP/WMI/Power Shell** and **Host Resources by Shell/SNMP/WMI/Power Shell**.

   c. **MS Cluster by NTCMD or UDA**

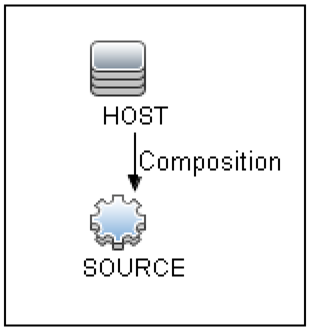
   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

MS Cluster by NTCMD or UDA Job

Input CI Type

Agent

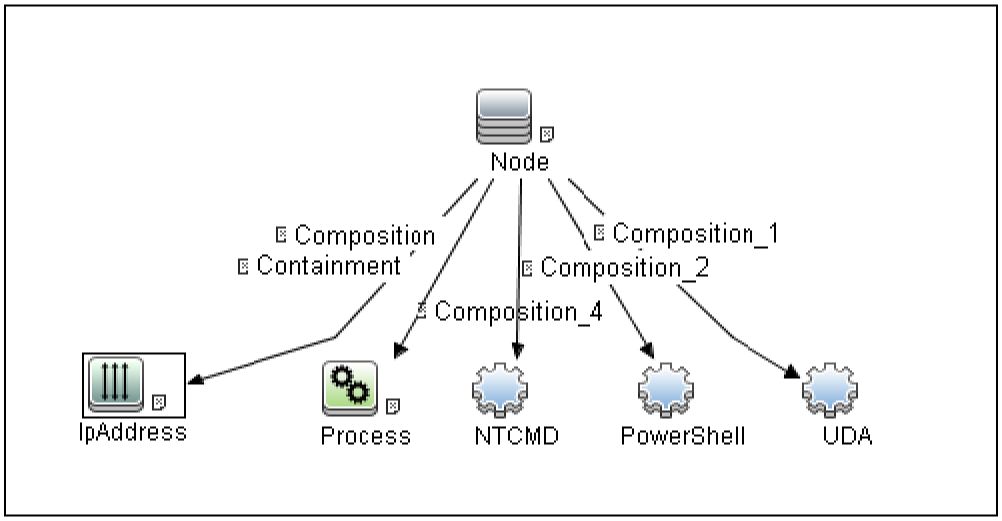
Input TQL Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Trigger Query



CI Attribute Conditions

CI	Condition
IpAddress	NOT IP Probe Name Is null
Process	Name Equal ignore case clussvc.exe
NTCMD	NOT Reference to the credentials dictionary entry Is null
PowerShell	NOT Reference to the credentials dictionary entry Is null

Used Scripts

- entity.py
- ms_cluster.py

- ms_cluster_discoverer.py
- MS_Cluster_Topology.py

Discovered CITs

For details on the CIs that are discovered, see the Statistics table in the **Details** tab.

- ClusterResourceGroup
- ClusterSoftware
- Composition
- ConfigurationDocument
- Containment
- Dependency
- ExecutionEnvironment
- IpAddress
- MS Cluster
- MSCS Resource Group
- MSCS resource
- Membership
- Node
- Ownership
- Virtual

Note: To view the topology, see "Microsoft Cluster Server View Topology" on page 311.

Chapter 23: Red Hat Cluster Suite Discovery

This chapter includes:

Overview

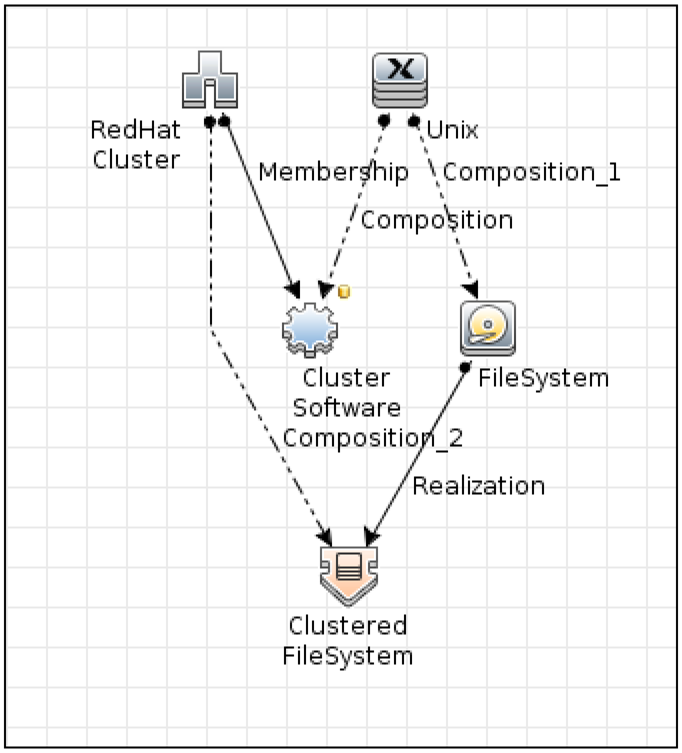
Red Hat Cluster Suite (RHCS) is a set of software components that enables setting up a high availability and load balancing cluster. Global File System 2 (GFS2) provides a clustered file system for use the with Red Hat Cluster Suite. GFS2 allows multiple nodes to share storage at a block level as if the storage is connected locally to each cluster node.

Supported Versions

Red Hat Cluster Suite Discovery is supported on Red Hat version 6.3.

Topology

The following diagram depicts the topology for Red Hat Cluster Suite Discovery.



How to Discover Red Hat Cluster Suite Topology

This section describes how to discover the topology managed by Red Hat Cluster Suite.

Prerequisites

- Ensure that there is shell connectivity to cluster nodes.
- Set up SSH protocol credentials.

Run the Discovery

Note: Red Hat Cluster Suite Discovery can be performed in shallow or deep mode. To perform a shallow discovery, carry out steps 1-3 below. To perform a deep discovery, carry out all of the steps below.

To perform Red Hat Cluster Suite Discovery, execute the following jobs in the listed order:

1. **Range IPs by ICMP** (discovers the target IPs)
2. **Host Connection by Shell** (discovers the target host and shell connectivity to it)
3. **Host Applications by Shell** (discovers Red Hat cluster software)

   Note: If you want to perform only a shallow discovery, stop here and do not continue to the next step.

4. **Red Hat Cluster by Shell** (discovers Red Hat cluster resources)

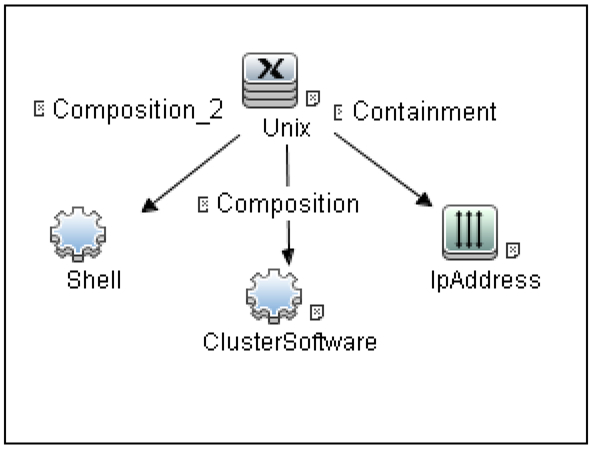
Red Hat Cluster by Shell Job

Adapter

ID: red_hat_cluster_by_shell

Trigger TQL

This job is triggered when **ClusterSoftware** has **ProductName=redhat_cluster** on the node with shell access, and the node's IP address is in the Probe's range. The shell should have valid credentials.



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
Shell	NOT Reference to the credentials dictionary entry Is null
Unix	None
ClusterSoftware	ProductName Equal redhat_cluster

Discovery Flow

The discovery flow for the **Red Hat Cluster by Shell** job is as follows:

1. The job obtains cluster information (**clustat -x**) and create the necessary cluster topology.

2. The job resolves hosts by using **nslookup** or the **hosts** file.

3. The job run the **blkid** command to determine if GFS/GFS2 is in use.

4. The job runs the **gfs_edit** and **blkid** commands to obtain the GFS configuration.

5. The job runs the **cat/etc/mtab** command to obtain mount points.

Red Hat Cluster by Shell Adapter

ID

red_hat_cluster_by_shell

Input CIT

Shell

Input TQL

The job is triggered when **ClusterSoftware** has **ProductName = redhat_cluster** and has a node with shell access, and the node's IP address is in the probe's range. The shell should have valid credentials.

Name	Value
Protocol	${SOURCE.root_class}
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Triggered CI Data

- **Protocol.** The shell's root_class.
- **CredentialsId.** The shell's credentials_id.
- **IP Address.** The shell's application_ip.

Used Script

- red_hat_cluster_by_shell.py

Discovered CITs

- ClusteredFileSystem
- ClusterSoftware
- Composition
- FileSystem
- Membership
- Node
- Realization
- RedHatCluster

Parameters

None

Limitations

Red Hat Cluster Suite Discovery has the following limitations:

- Currently, only GFS and basic cluster information discovery is supported.
- Clustered services discovery is not supported.
- LVM discovery is not supported.

Chapter 24: HP Serviceguard Cluster Discovery

This chapter includes:

Overview

HP Serviceguard is the cluster solution for HP-UX. HP Global Workload Management adjusts workloads to optimize performance, and integrates with Instant Capacity on Demand. HP Serviceguard allows the clustering of FS with the installed services. The **Service Guard Cluster Topology by TTY** job discovers CIs like packages, file system elements, and running services, with the corresponding logical links.

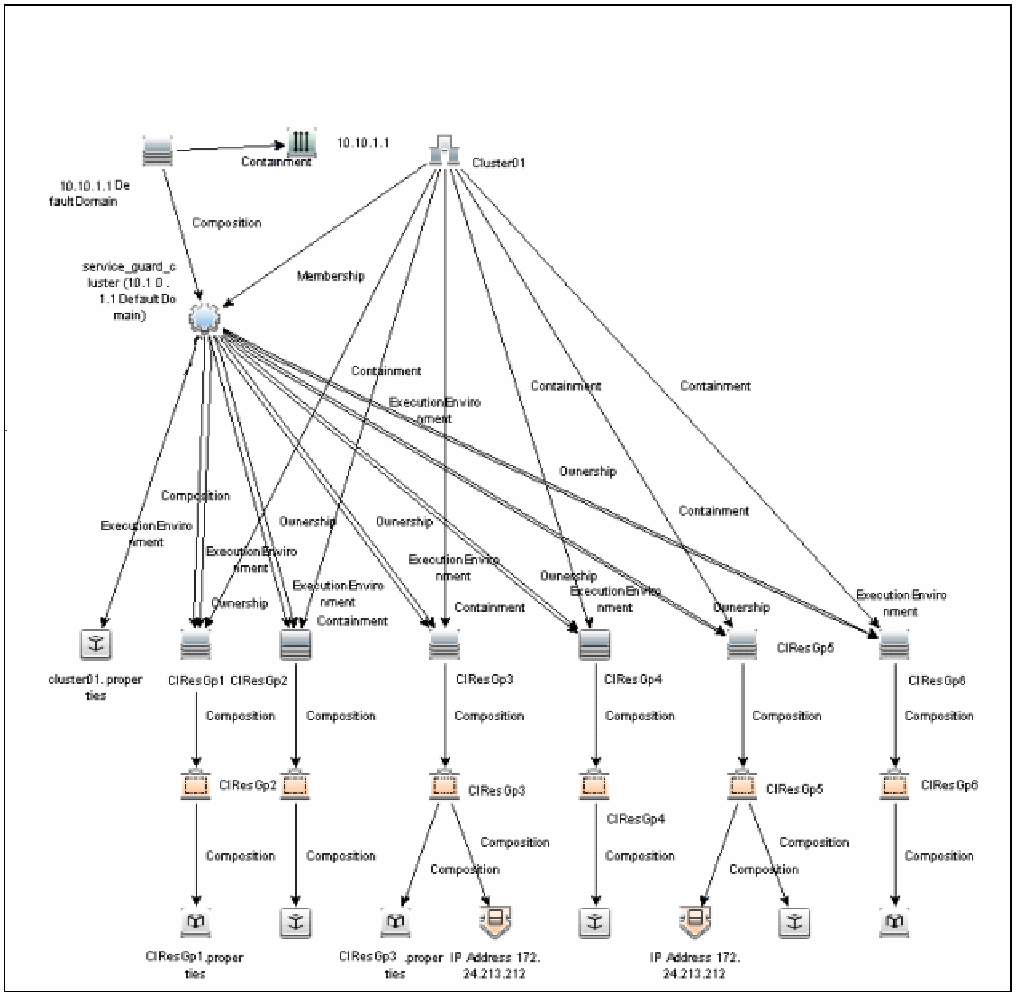
Supported Versions

This discovery solution supports HP Serviceguard Cluster on top of HP-UX 10.xx and 11.xx.

Topology

The following image displays the topology of the HP Serviceguard Cluster Discovery.

Note: For a list of discovered CITs, see "Service Guard Cluster Topology Adapter" on page 327.



How to Discover HP Serviceguard Cluster Topology

This task explains how to discover Serviceguard Cluster Topology.

1. **Prerequisite - Permissions**

   Before starting the discovery, ensure the user has the permissions required to run the following commands:

   - `/usr/sbin/cmviewcl -v`
   - `cat <package config or log>`
   - `uname`

- `ps -ef`
- `lsnrctl status`
- `pfiles`
- `lsof`

2. **Prerequisite - Set up protocol credentials**

   To discover HP Serviceguard cluster topology, you must set up the appropriate Shell protocol: SSH, Telnet, or both, depending on the particular system being accessed. Prepare the following information for the Shell protocol: **user name**, **password**, and **domain name**.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

3. **Run the discovery**

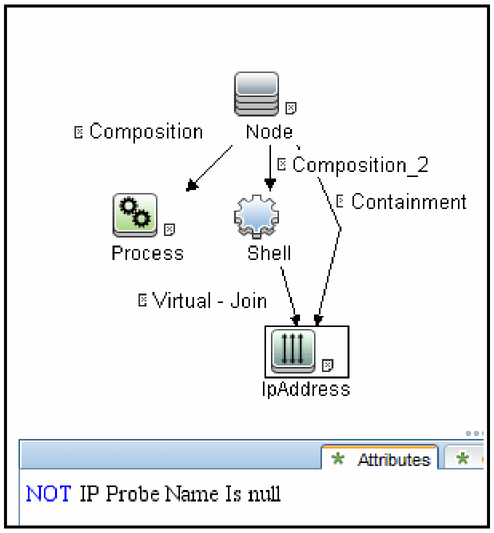
   Run the following jobs:

   - **Range IPs by ICMP** to discover the HP Serviceguard cluster IP addresses
   - **Host Connection by Shell** to discover the HP Serviceguard system with the SSH agent and networking topology connected
   - **Host Applications by Shell** to discover if HP Serviceguard is set up and running on the destination
   - **Service Guard Cluster Topology by TTY**

For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Service Guard Cluster Topology by TTY Job

Trigger Query



NOT IP Probe Name Is null

CIT	Attribute
IpAddress	NOT IP Probe Name Is null
Shell	None
Node	None
Process	Name Equal ignore case cmcld OR Name Equal ignore case cmclconfd

Adapter

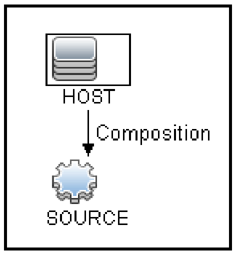
This job uses the **Service Guard Cluster Topology** adapter.

For details, see "Service Guard Cluster Topology Adapter" on the next page.

Service Guard Cluster Topology Adapter

This section contains details about the adapter.

Input Query



Input CIT

Shell

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
cmclconfig_path	${PARAMETERS.cmclconfig_path}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Used Scripts

- file_mon_utils.py
- file_ver_lib.py
- oracle_shell_utils.py
- service_guard.py

- Service_Guard_Cluster_Topology.py
- service_guard_discoverers.py
- service_guard_topology.py

Discovered CITs

- Clustered Software
- Composition
- ConfigurationDocument
- Containment Dependency
- ExecutionEnvironment
- ipAddress Membership
- Node
- Ownership
- SG Package
- SG Resource
- Service Guard Cluster

HP Serviceguard Cluster Commands

This section includes the Serviceguard clustering commands.

`/usr/local/bin/sudo /usr/sbin/cmviewcl -v`

Output:

```
CLUSTER          STATUS
SomeClusterName    up

NODE    STATUS    STATE
Node1    up    running
```

```
Quorum_Server_Status:
NAME       STATUS    STATE
172.24.0.5 up       running

 Network_Parameters:
INTERFACE   STATUS   PATH              NAME
PRIMARY     up     0/2/2/1          lan3
PRIMARY     up     0/1/1/1            lan1
PRIMARY     up     0/2/2/0            lan2
PRIMARY     up    0/1/1/0            lan0
STANDBY     up     0/3/0/0/0/0/4/0/0/ lan7
STANDBY     up     0/3/0/0/0/0/2/0/0/ lan5
STANDBY     up    0/3/0/0/0/0/4/0/0/ lan6
STANDBY     up                      0/3/0/0/0/0/2/0/0/ lan4

PACKAGE   STATUS   STATE   AUTO_RUN   NODE
PackageName1 up    running     enabled   Node1

 Policy_Parameters:
POLICY_NAME CONFIGURED_VALUE
Failover   configured_node
Failback   manual

 Node_Switching_Parameters:
NODE_TYPE  STATUS  SWITCHING  NAME
Primary    up    enabled Node1 (current)

PACKAGE   STATUS   STATE   AUTO_RUN   NODE
 PackageName2  up running enabled Node1

 Policy_Parameters:
POLICY_NAME CONFIGURED_VALUE
Failover   configured_node
Failback   manual

 Script_Parameters:
ITEM   STATUS   MAX_RESTARTS RESTARTS NAME
Subnet   up     192.168.62.0
Subnet   up     172.24.0.0

 Node_Switching_Parameters:
NODE_TYPE  STATUS  SWITCHING NAME
Primary      up    enabled Node1 (current)
```

```
PACKAGE STATUS STATE AUTO_RUN NODE
PackageName3 up running enabled Node1

 Policy_Parameters:

POLICY_NAME CONFIGURED_VALUE
Failover configured_node
Failback manual

 Node_Switching_Parameters:
NODE_TYPE STATUS SWITCHING NAME
Primary    up    enabled Node1 (current)
```

Mapping

Output of this command is used to fill in the attributes of the CIs:

CMD Output Attribute	CI Name	CI Attribute
SomeClusterName	Service Guard Cluster	Name
PackageName1,..., PackageName3	SG Package	Name
IP Address	SG Resource	Name
Subnet value	Network	Name
Node1	Node	Name

`find /etc/cmcluster/ -name '*.cfg'`

Output:

```
/etc/cmcluster/scripts/exampleapplicatie.cfg
/etc/cmcluster/package1/package1.cfg
/etc/cmcluster/package2/package2.cfg
/etc/cmcluster/package3/package3.cfg
```

Mapping:

This command is used to find package configuration files in the SG Cluster configuration directory.

find /etc/cmcluster/ -name '*.config'

Output:

```
/etc/cmcluster/scripts/exampleapplicatie.config
/etc/cmcluster/package1/package1.config
/etc/cmcluster/package2/package2.config
/etc/cmcluster/package3/package3.config
```

Mapping:

This command is used to find package configuration files in the SG Cluster configuration directory.

cat "/etc/cmcluster/package1/package.cfg" | grep -iE "PACKAGE_NAME|SCRIPT_LOG_FILE|RUN_SCRIPT|FS_DIRECTORY"

Output:

```
# "PACKAGE_NAME" is the name that is used to identify the package.
# Legal values for PACKAGE_NAME:
PACKAGE_NAME package1
# "RUN_SCRIPT" is the script that starts a package.
# Legal values for RUN_SCRIPT:
RUN_SCRIPT /etc/cmcluster/package1/package1.cntl
# "RUN_SCRIPT_TIMEOUT" is the number of seconds allowed for the package to start.
```

Mapping:

PACKAGE_NAME and RUN_SCRIPT variable values are used in further commands for discovery of IP and Mount Points, which are managed by this package.

cat "/etc/cmcluster/package1/package1.cntl.log" | grep -E "Mounting"

Output:

```
Jul 11 09:27:10 - Node "Node1":
 Mounting /dev/vg1/lvol1 at /oracle/somename1
 Jul 11 09:27:22 - Node "Node1":
 Mounting /dev/vg1/lvol2 at /oracle/somename2
 Jul 11 09:27:53 - Node "Node1":
 Mounting /dev/vg1/lvol3 at /oracle/somename3
```

Mapping:

Discovered data for mount points will be used to link the RunningSoftware to the proper Clustered Service (actually a package). This linking approach relies on the running process path.

`cat "/etc/cmcluster/package1/package1.cntl.log" | grep -E "Adding IP"`

Output:

```
Jun 12 09:27:11 - Node "Node1":
 Adding IP address 192.168.62.146 to subnet 192.168.62.0
 Jun 12 09:27:11 - Node "Node1":
 Adding IP address 172.24.10.142 to subnet 172.24.0.0
```

Mapping:

Discovered IP Address and Network will be reported as corresponding CIs. This is done because not all IP Resources might be present in the cmviewcl output.

`ps -ef | grep "tnslsnr"`

Output:

```
orauser 21926 1 0 Jun 9 ?
 6:09 /oracle/somename1/applic/oracle/db/
 10.2.0/instns1/bin/tnslsnr listener_name1 -inherit
```

Mapping:

From the fetched Oracle Listener process information ORACLE_HOME value, listener name and pid will be parsed out. ORACLE_HOME and listener name will be used in further discovery to get listener status and parse out Oracle DB SIDs.

```
/oracle/somename1/applic/oracle/db/10.2.0/instns1/bin/lsnrctl
status listener_name1
```

Output:

```
LSNRCTL for HPUX: Version 10.2.0.5.0 -
Production on 20-JUL-2011 06:44:11
Copyright (c) 1991, 2010, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=oic6)))
STATUS of the LISTENER
------------------------
Alias listener_name1
Version TNSLSNR for HPUX: Version 10.2.0.5.0 -
 Production Start Date 09-JUN-2011 21:56:34
Uptime 40 days 8 hr. 47 min. 36 sec
Trace Level off
Security ON: Local OS Authentication
SNMP OFF
Listener Parameter File
/oracle/somename1/applic/oracle/db/10.2.0/
instns1/network/admin/listener.ora
Listener Log File
/oracle/somename1/applic/oracle/db/10.2.0/
instns1/network/log/listener_name1.log
Listening Endpoints Summary...
DESCRIPTION=
ADDRESS=(PROTOCOL=ipc)(KEY=sid1)))
DESCRIPTION=
ADDRESS=(PROTOCOL=tcp)(HOST=192.168.80.24)(PORT=1521)))
DESCRIPTION=
ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROCsidas)))

Services Summary...
Service "PLSExtProcsid1" has 1 instance(s).
Instance "PLSExtProcdis1", status UNKNOWN,
has 1 handler(s) for this service...
Service "sid1.somedomain" has 1 instance(s).
Instance "sid1", status UNKNOWN,
has 1 handler(s) for this service...
The command completed successfully
```

Mapping:

Instance value will be parsed out and treated as SID; any instance name starting from PLSExtProc will be filtered out since this is RPC call service.

nice lsof -i 4 -a -P -n -p <Actual Pid> **or** nice pfiles 21926 2>&1 | awk "/S_IFSOCK|SOCK_STREAM|SOCK_DGRAM|port/ { print }"

Mapping:

Discovered IP and port information is used to set Application IP and Port on reported Running Software.

cat "/oracle/somename2/applic/oracle/oas/10.1.2/somename5/opmn/conf/opmn.xml"

Output:

```
......skip......
<ias-instance id="somename.somedomain">
<environment>
<variable id="TMP" value="/tmp"/>
<variable id="LD_LIBRARY_PATH" value="/usr/lib"/>
<variable id="LD_PRELOAD" value="libloghost.so.1"/>
......skip......
```

Mapping:

The Oracle iAS CI name is taken from value of ias-instance in the following order: parameter name, parameter id, Default Server.

Chapter 25: Solaris Cluster Discovery

This chapter includes:

Overview

The Sun Cluster product is an integrated hardware and software solution used to create highly available and scalable services. The Sun Cluster environment extends the Solaris Operating System into a cluster operating system. A cluster is a collection of one or more nodes that belong exclusively to that collection.

Supported Versions

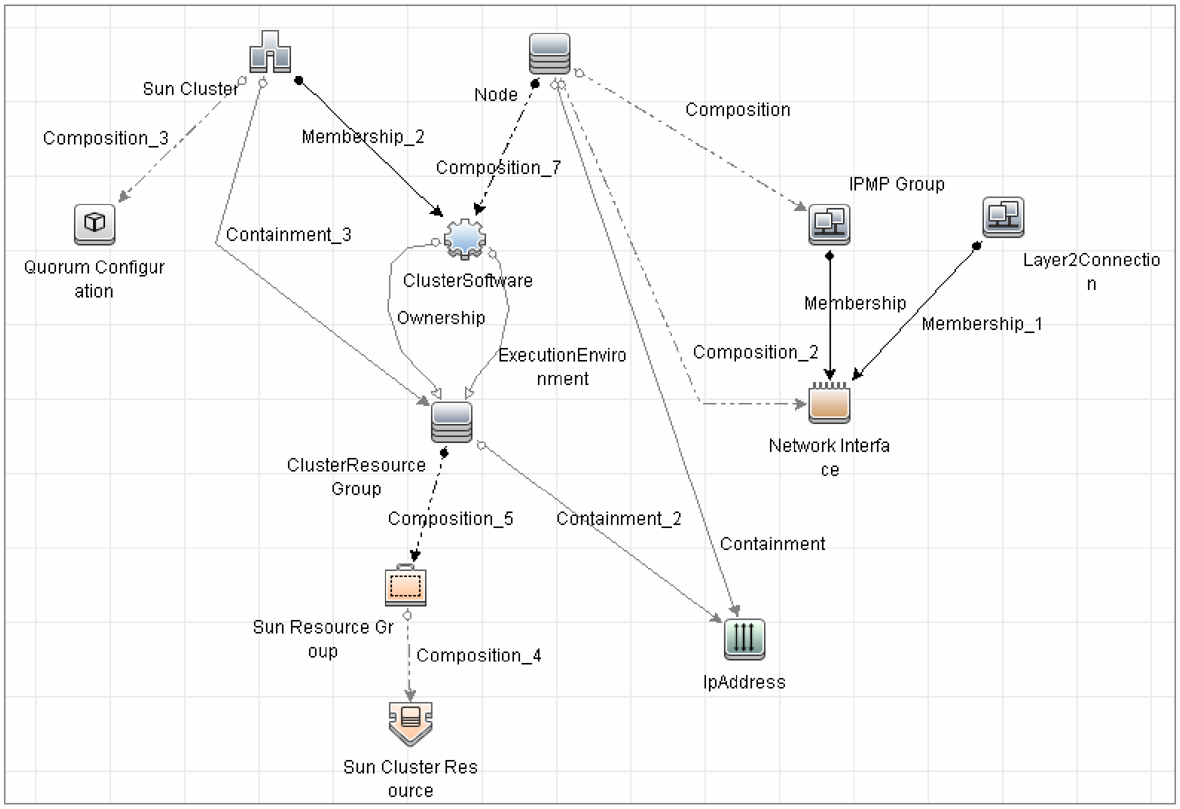
The Sun Cluster package supports Sun Cluster 3.2. Support for older versions of Sun Cluster has not been verified.

The Sun Cluster software integrates with the Solaris operating system, thus only this operating system is supported.

Topology

The following image displays the topology of the Solaris Cluster discovery.

Note: For a list of discovered CITs, see "Sun Cluster by Shell Job" on page 338.


Sun Cluster
Node
Composition_3
Membership_2
Composition
Composition_7
IPMP Group
Quorum Configur ation
Containment_3
ClusterSoftware
Layer2Connectio n
Membership
Membership_1
Ownership
ExecutionEnviro nment
Composition_2
Network Interfa ce
ClusterResource Group
Composition_5
Containment_2
Containment
Sun Resource Gr oup
Composition_4
IpAddress
Sun Cluster Res ource

How to Discover Sun Cluster

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials and permissions**

   - This discovery uses the Telnet and SSH protocols.

     For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   - Set up permissions for users performing Solaris Cluster discovery to run clustering commands (**scrgadm**, **scstat**, **scconf**, and so on). For a full list of commands see "Solaris Cluster Discovery Commands" on page 342.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Run the following jobs in the following order:

   a. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   b. Run the **Host Connection by Shell** job to discover Shell connectivity and basic information about the hosts.

   c. Run the **Host Applications by Shell** job to discover processes on the target machines.

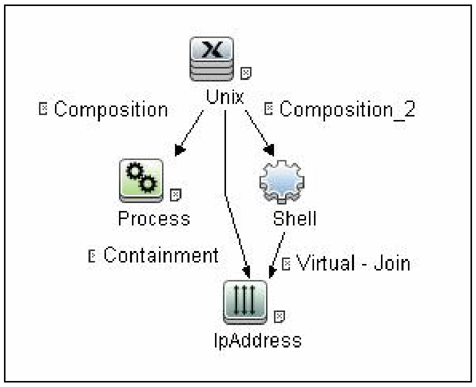
   d. Run the **Sun Cluster by Shell** job to discover the Sun Cluster topology. For job details, see "Sun Cluster by Shell Job" below.

Sun Cluster by Shell Job

Adapter

ID:`Sun_Cluster_by_Shell`

Trigger TQL



CI Attribute Conditions

Attribute	Condition
Process	`Name Equal ignore case "cluster"`
Shell	`NOT Reference to the credentials dictionary entry is null`
IpAddress	`Not IP Probe Name is null`

Parameters

None

Prerequisites

See the prerequisites in "How to Discover Sun Cluster" on the previous page.

Discovery Flow

The discovery flow for the Sun Cluster by Shell job is as follows:

1. Get the cluster configuration (using the command **/usr/cluster/bin/scconf -pv**), including:
   - Cluster name
   - Names of cluster nodes
   - Transport adapters
   - Quorum devices

2. Get the cluster version (using the command **/usr/cluster/bin/scinstall –p**).

3. Get the quorum status (using the command **/usr/cluster/bin/scstat -q**).

4. Analyze the cluster nodes that were found and resolve their hostnames to IPs.

5. Get the statuses of the cluster nodes (using the command **/usr/cluster/bin/scstat -n**).

6. Get the standard network information, specifically **netstat** information (using the command **/usr/bin/netstat -np**). This information is used to resolve MAC addresses of transport adapters on the cluster node that the job connected to.

7. Resolve IP addresses of transport adapters to their MACs via the ARP table for cluster nodes that are not the node the job is currently connected to (using the command **/usr/sbin/arp <IP>**).

8. Get cluster resources (using the command **/usr/cluster/bin/scrgadm –pvv**), including:

   - Resource Groups

   - Resources. Host names in resources of types **LogicalHostname** or **SharedAddress** are resolved to IPs.

9. Get statuses of resource groups (using the command **/usr/cluster/bin/scstat -g**).

10. Get transport paths (using the command **/usr/cluster/bin/scstat –W**).

11. Create the result vector with the topology discovered and send it to the UCMDB.

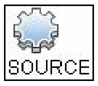
Sun Cluster Adapter

Input CIT

Shell

Input Query

This query contains only one Shell CI:



Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Used Scripts

- networking.py
- solaris_networking.py
- sun_cluster_by_shell.py

Discovered CITs

- ClusterSoftware
- Composition
- ConfigurationDocument
- Containment
- ExecutionEnvironment
- Interface
- IpAddress
- Layer2Connection
- Membership
- Node
- Sun Cluster

- Sun Cluster Resource
- Sun Resource Group

Note: To view the topology, see "Topology" on page 336.

Solaris Cluster Discovery Commands

This section includes the Sun clustering commands:

- "Get Name of Cluster" below
- "Get Nodes of Cluster" on page 344
- "Resolve Node Names to IPs" on page 344
- "Get Status of Nodes" on page 345
- "Get Resource Groups and Resources" on page 345
- "Get Details for Resource Groups and Resources" on page 346
- "Get Cluster Interconnection Information" on page 357
- "Get Quorum Configuration" on page 360

Get Name of Cluster

Command	/usr/cluster/bin/scconf -p

Example of output	`Cluster name: cluster1`
	
	`Cluster ID: 0x4A7BDCD3`
	
	`Cluster install mode: disabled`
	
	`Cluster private net: 172.2.0.0`
	
	`Cluster private netmask: 255.255.255.192`
	
	`Cluster maximum nodes: 6`
	
	`Cluster maximum private networks: 4`
	
	`Cluster new node authentication: unix`
	
	`Cluster authorized-node list: <. - Exclude all nodes>`
	
	`Cluster transport heart beat timeout: 10000`
	
	`Cluster transport heart beat quantum: 1000`
	
	`Round Robin Load Balancing UDP session timeout: 480`
	
	`Cluster nodes: node1 node2`
	
	`Cluster node name: node1`
	
	`...`

Values taken	Name of the cluster: cluster1
Comments	Name of the cluster enabling the creation of the Sun Cluster CI.

Get Nodes of Cluster

Command	/usr/cluster/bin/scconf -p

Example of output	`Cluster name: cluster1`
	`Cluster ID: 0x4A7BDCD3`
	`Cluster install mode: disabled`
	`Cluster private net: 172.2.7.0`
	`Cluster private netmask: 255.255.255.192`
	`Cluster maximum nodes: 6`
	`Cluster maximum private networks: 4`
	`Cluster new node authentication: unix`
	`Cluster authorized-node list: <. - Exclude all nodes>`
	`Cluster transport heart beat timeout: 10000`
	`Cluster transport heart beat quantum: 1000`
	`Round Robin Load Balancing UDP session timeout: 480`
	`Cluster nodes: node1 node2`
	`...`

Values taken	Node names

Resolve Node Names to IPs

Command	/usr/sbin/nslookup node1

Example of output	`Server: 134.44.0.10`
	`Address: 134.44.0.10#53`
	`Name: node1.example.com`
	`Address: 134.44.0.75`

Values taken	IP of the node
Comments	The IP enables the creation of an incomplete Host for each node in the cluster

Get Status of Nodes

Command	/usr/cluster/bin/scstat -n

Example of output	`-- Cluster Nodes --`
	` Node name Status`
	` --------- ------`
	` Cluster node: node1 Online`
	` Cluster node: node2 Online`

Values taken	Node statuses
Comments	Although statuses are not reported, Discovery needs this status. For example, Discovery should not issue an arp command to resolve the MAC address if the node is off-line.

Get Resource Groups and Resources

Command	/usr/cluster/bin/scstat -g

Example of output	`-- Resource Groups and Resources --`
	` Group Name Resources`
	` ---------- ---------`
	` Resources: oracle1 oracle1-zfs oracle1-lh oracle1-ora oracle1-cron oracle1-lsnr_ano_10`
	
	`-- Resource Groups --`
	`...`

Values taken	List of groups. List of resources in a group. Status of a group on each of the nodes (run links are created based on this).

Get Details for Resource Groups and Resources

Command	/usr/cluster/bin/scrgadm -pvv

Example of output	
	```
	Res Group name: oracle1
	 (oracle1) Res Group RG_description: <NULL>
	 (oracle1) Res Group mode: Failover
	 (oracle1) Res Group management state: Managed
	 (oracle1) Res Group RG_project_name: user.oracle
	 (oracle1) Res Group RG_SLM_type: manual
	 (oracle1) Res Group RG_affinities: <NULL>
	 (oracle1) Res Group Auto_start_on_new_cluster: True
	 (oracle1) Res Group Failback: False
	 (oracle1) Res Group Nodelist: node1 node2
	 (oracle1) Res Group Maximum_primaries: 1
	 (oracle1) Res Group Desired_primaries: 1
	 (oracle1) Res Group RG_dependencies: <NULL>
	 (oracle1) Res Group network dependencies: True
	 (oracle1) Res Group Global_resources_used: <All>
	 (oracle1) Res Group Pingpong_interval: 3600
	 (oracle1) Res Group Pathprefix: <NULL>
	 (oracle1) Res Group system: False
	 (oracle1) Res Group Suspend_automatic_recovery: False
	(oracle1) Res name: oracle1-zfs
	(oracle1:oracle1-zfs) Res R_description:
	(oracle1:oracle1-zfs) Res resource type: SUNW.HAStoragePlus:8
	(oracle1:oracle1-zfs) Res type version: 8
	(oracle1:oracle1-zfs) Res resource group name: oracle1
	(oracle1:oracle1-zfs) Res resource project name: user.oracle
	 (oracle1:oracle1-zfs{kvsdb1}) Res enabled: True
	```

Command	/usr/cluster/bin/scrgadm -pvv

Example of output (continued)	`(oracle1:oracle1-zfs{kvsdb2}) Res enabled: True`
	
	`(oracle1:oracle1-zfs{kvsdb1}) Res monitor enabled: True`
	
	`(oracle1:oracle1-zfs{kvsdb2}) Res monitor enabled: True`
	
	`(oracle1:oracle1-zfs) Res strong dependencies: <NULL>`
	
	`(oracle1:oracle1-zfs) Res weak dependencies: <NULL>`
	
	`(oracle1:oracle1-zfs) Res restart dependencies: <NULL>`
	
	`(oracle1:oracle1-zfs) Res offline restart dependencies: <NULL>`
	
	`(oracle1:oracle1-zfs) Res property name: Retry_interval`
	
	`(oracle1:oracle1-zfs:Retry_interval) Res property class: standard`
	
	`(oracle1:oracle1-zfs:Retry_interval) Res property description: Time in which monitor attempts to restart a failed resource Retry_count times.`
	
	`(oracle1:oracle1-zfs:Retry_interval) Res property type: int`
	
	`(oracle1:oracle1-zfs:Retry_interval) Res property value: 300`
	
	`(oracle1:oracle1-zfs) Res property name: Retry_count`
	
	`(oracle1:oracle1-zfs:Retry_count) Res property class: standard`
	
	`(oracle1:oracle1-zfs:Retry_count) Res property description: Indicates the number of times a monitor restarts the resource if it fails.`
	
	`(oracle1:oracle1-zfs:Retry_count) Res property type: int`
	
	`(oracle1:oracle1-zfs:Retry_count) Res property value: 2`
	
	`(oracle1:oracle1-zfs) Res property name: Failover_mode`
	
	`(oracle1:oracle1-zfs:Failover_mode) Res property class: standard`
	
	`(oracle1:oracle1-zfs:Failover_mode) Res property description: Modifies recovery actions taken when the resource fails.`
	
	`(oracle1:oracle1-zfs:Failover_mode) Res property type: enum`
	
	`(oracle1:oracle1-zfs:Failover_mode) Res property value: SOFT`
	
	`(oracle1:oracle1-zfs) Res property name: POSTNET_STOP_TIMEOUT`
	
	`(oracle1:oracle1-zfs:POSTNET_STOP_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-zfs:POSTNET_STOP_TIMEOUT) Res property description: Maximum execution time allowed for Postnet_stop`

Command	/usr/cluster/bin/scrgadm -pvv

	`method.`
	
	` (oracle1:oracle1-zfs:POSTNET_STOP_TIMEOUT) Res property type: int`
	
	` (oracle1:oracle1-zfs:POSTNET_STOP_TIMEOUT) Res property value: 1800`
	
	`(oracle1:oracle1-zfs) Res property name: PRENET_START_TIMEOUT`
	
	` (oracle1:oracle1-zfs:PRENET_START_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-zfs:PRENET_START_TIMEOUT) Res property description: Maximum execution time allowed for Prenet_Start method.`
	
	` (oracle1:oracle1-zfs:PRENET_START_TIMEOUT) Res property type: int`
	
	` (oracle1:oracle1-zfs:PRENET_START_TIMEOUT) Res property value: 1800`
	
	` (oracle1:oracle1-zfs) Res property name: MONITOR_CHECK_TIMEOUT`
	
	`(oracle1:oracle1-zfs:MONITOR_CHECK_TIMEOUT) Res property class: standard`
	
	` (oracle1:oracle1-zfs:MONITOR_CHECK_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Check method.`
	
	` (oracle1:oracle1-zfs:MONITOR_CHECK_TIMEOUT) Res property type: int`
	
	` (oracle1:oracle1-zfs:MONITOR_CHECK_TIMEOUT) Res property value: 90`
	
	` (oracle1:oracle1-zfs) Res property name: MONITOR_STOP_TIMEOUT`
	
	`(oracle1:oracle1-zfs:MONITOR_STOP_TIMEOUT) Res property class: standard`
	
	` (oracle1:oracle1-zfs:MONITOR_STOP_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Stop method.`
	
	`(oracle1:oracle1-zfs:MONITOR_STOP_TIMEOUT) Res property type: int (oracle1:oracle1-zfs:MONITOR_STOP_TIMEOUT) Res property value: 90`
	
	`(oracle1:oracle1-zfs) Res property name: MONITOR_START_TIMEOUT`
	
	`(oracle1:oracle1-zfs:MONITOR_START_TIMEOUT) Res property class: standard`

Command	/usr/cluster/bin/scrgadm -pvv
	(oracle1:oracle1-zfs:MONITOR_START_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Start method. (oracle1:oracle1-zfs:MONITOR_START_TIMEOUT) Res property type: int (oracle1:oracle1-zfs:MONITOR_START_TIMEOUT) Res property value: 90 (oracle1:oracle1-zfs) Res property name: INIT_TIMEOUT (oracle1:oracle1-zfs:INIT_TIMEOUT) Res property class: standard (oracle1:oracle1-zfs:INIT_TIMEOUT) Res property description: Maximum execution time allowed for Init method. (oracle1:oracle1-zfs:INIT_TIMEOUT) Res property type: int (oracle1:oracle1-zfs:INIT_TIMEOUT) Res property value: 1800 (oracle1:oracle1-zfs) Res property name: UPDATE_TIMEOUT (oracle1:oracle1-zfs:UPDATE_TIMEOUT) Res property class: standard (oracle1:oracle1-zfs:UPDATE_TIMEOUT) Res property description: Maximum execution time allowed for Update method. (oracle1:oracle1-zfs:UPDATE_TIMEOUT) Res property type: int (oracle1:oracle1-zfs:UPDATE_TIMEOUT) Res property value: 1800 (oracle1:oracle1-zfs) Res property name: VALIDATE_TIMEOUT (oracle1:oracle1-zfs:VALIDATE_TIMEOUT) Res property class: standard (oracle1:oracle1-zfs:VALIDATE_TIMEOUT) Res property description: Maximum execution time allowed for Validate method. (oracle1:oracle1-zfs:VALIDATE_TIMEOUT) Res property type: int (oracle1:oracle1-zfs:VALIDATE_TIMEOUT) Res property value: 1800 (oracle1:oracle1-zfs) Res property name: ZpoolsSearchDir (oracle1:oracle1-zfs:ZpoolsSearchDir) Res property class: extension (oracle1:oracle1-zfs:ZpoolsSearchDir) Res property description: Directory location to search devices for zpools (oracle1:oracle1-zfs:ZpoolsSearchDir) Res property pernode: False

Command	/usr/cluster/bin/scrgadm -pvv
	(oracle1:oracle1-zfs:ZpoolsSearchDir) Res property type: string (oracle1:oracle1-zfs:ZpoolsSearchDir) Res property value: (oracle1:oracle1-zfs) Res property name: FilesystemCheckCommand (oracle1:oracle1-zfs:FilesystemCheckCommand) Res property class: extension (oracle1:oracle1-zfs:FilesystemCheckCommand) Res property description: Command string to be executed for file system checks (oracle1:oracle1-zfs:FilesystemCheckCommand) Res property pernode: False (oracle1:oracle1-zfs:FilesystemCheckCommand) Res property type: stringarray (oracle1:oracle1-zfs:FilesystemCheckCommand) Res property value: <NULL> (oracle1:oracle1-zfs) Res property name: AffinityOn (oracle1:oracle1-zfs:AffinityOn) Res property class: extension (oracle1:oracle1-zfs:AffinityOn) Res property description: For specifying affinity switchover (oracle1:oracle1-zfs:AffinityOn) Res property pernode: False (oracle1:oracle1-zfs:AffinityOn) Res property type: boolean (oracle1:oracle1-zfs:AffinityOn) Res property value: TRUE (oracle1:oracle1-zfs) Res property name: FilesystemMountPoints (oracle1:oracle1-zfs:FilesystemMountPoints) Res property class: extension (oracle1:oracle1-zfs:FilesystemMountPoints) Res property description: The list of file system mountpoints (oracle1:oracle1-zfs:FilesystemMountPoints) Res property pernode: False (oracle1:oracle1-zfs:FilesystemMountPoints) Res property type: stringarray (oracle1:oracle1-zfs:FilesystemMountPoints) Res property value: <NULL> (oracle1:oracle1-zfs) Res property name: GlobalDevicePaths (oracle1:oracle1-zfs:GlobalDevicePaths) Res property class: extension

Command	/usr/cluster/bin/scrgadm -pvv

```
(oracle1:oracle1-zfs:GlobalDevicePaths) Res property description:
The list of HA global device paths

(oracle1:oracle1-zfs:GlobalDevicePaths) Res property pernode:
False

 (oracle1:oracle1-zfs:GlobalDevicePaths) Res property type:
stringarray

(oracle1:oracle1-zfs:GlobalDevicePaths) Res property value:
<NULL>

 (oracle1:oracle1-zfs) Res property name: Zpools

(oracle1:oracle1-zfs:Zpools) Res property class: extension

 (oracle1:oracle1-zfs:Zpools) Res property description: The list
of zpools

 (oracle1:oracle1-zfs:Zpools) Res property pernode: False

 (oracle1:oracle1-zfs:Zpools) Res property type: stringarray

 (oracle1:oracle1-zfs:Zpools) Res property value: oracle1prod

(oracle1) Res name: oracle1-lh

 (oracle1:oracle1-lh) Res R_description:

 (oracle1:oracle1-lh) Res resource type: SUNW.LogicalHostname:2

 (oracle1:oracle1-lh) Res type version: 2

(oracle1:oracle1-lh) Res resource group name: oracle1

 (oracle1:oracle1-lh) Res resource project name: user.oracle

(oracle1:oracle1-lh{kvsdb1}) Res enabled: True

 (oracle1:oracle1-lh{kvsdb2}) Res enabled: True

 (oracle1:oracle1-lh{kvsdb1}) Res monitor enabled: True

 (oracle1:oracle1-lh{kvsdb2}) Res monitor enabled: True

(oracle1:oracle1-lh) Res strong dependencies: <NULL>

 (oracle1:oracle1-lh) Res weak dependencies: <NULL>

(oracle1:oracle1-lh) Res restart dependencies: <NULL>

(oracle1:oracle1-lh) Res offline restart dependencies: <NULL>

(oracle1:oracle1-lh) Res property name: Retry_interval

 (oracle1:oracle1-lh:Retry_interval) Res property class: standard
```

Command	/usr/cluster/bin/scrgadm -pvv
	(oracle1:oracle1-lh:Retry_interval) Res property description: Time in which monitor attempts to restart a failed resource Retry_count times. (oracle1:oracle1-lh:Retry_interval) Res property type: int (oracle1:oracle1-lh:Retry_interval) Res property value: 300 (oracle1:oracle1-lh) Res property name: Retry_count (oracle1:oracle1-lh:Retry_count) Res property class: standard (oracle1:oracle1-lh:Retry_count) Res property description: Indicates the number of times a monitor restarts the resource if it fails. (oracle1:oracle1-lh:Retry_count) Res property type: int (oracle1:oracle1-lh:Retry_count) Res property value: 2 (oracle1:oracle1-lh) Res property name: Thorough_probe_interval (oracle1:oracle1-lh:Thorough_probe_interval) Res property class: standard (oracle1:oracle1-lh:Thorough_probe_interval) Res property description: Time between invocations of a high-overhead fault probe of the resource. (oracle1:oracle1-lh:Thorough_probe_interval) Res property type: int (oracle1:oracle1-lh:Thorough_probe_interval) Res property value: 60 (oracle1:oracle1-lh) Res property name: Cheap_probe_interval (oracle1:oracle1-lh:Cheap_probe_interval) Res property class: standard (oracle1:oracle1-lh:Cheap_probe_interval) Res property description: Time between invocations of a quick fault probe of the resource. (oracle1:oracle1-lh:Cheap_probe_interval) Res property type: int (oracle1:oracle1-lh:Cheap_probe_interval) Res property value: 60 (oracle1:oracle1-lh) Res property name: Failover_mode (oracle1:oracle1-lh:Failover_mode) Res property class: standard (oracle1:oracle1-lh:Failover_mode) Res property description: Modifies recovery actions taken when the resource fails.

Command	/usr/cluster/bin/scrgadm -pvv

	`(oracle1:oracle1-lh:Failover_mode) Res property type: enum`
	
	`(oracle1:oracle1-lh:Failover_mode) Res property value: HARD`
	
	`(oracle1:oracle1-lh) Res property name: PRENET_START_TIMEOUT`
	
	`(oracle1:oracle1-lh:PRENET_START_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:PRENET_START_TIMEOUT) Res property description: Maximum execution time allowed for Prenet_Start method.`
	
	`(oracle1:oracle1-lh:PRENET_START_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:PRENET_START_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: MONITOR_CHECK_TIMEOUT`
	
	`(oracle1:oracle1-lh:MONITOR_CHECK_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:MONITOR_CHECK_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Check method.`
	
	`(oracle1:oracle1-lh:MONITOR_CHECK_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:MONITOR_CHECK_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: MONITOR_STOP_TIMEOUT`
	
	`(oracle1:oracle1-lh:MONITOR_STOP_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:MONITOR_STOP_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Stop method.`
	
	`(oracle1:oracle1-lh:MONITOR_STOP_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:MONITOR_STOP_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: MONITOR_START_TIMEOUT`
	
	`(oracle1:oracle1-lh:MONITOR_START_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:MONITOR_START_TIMEOUT) Res property description: Maximum execution time allowed for Monitor_Start method.`

Command	/usr/cluster/bin/scrgadm -pvv

	`(oracle1:oracle1-lh:MONITOR_START_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:MONITOR_START_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: UPDATE_TIMEOUT`
	
	`(oracle1:oracle1-lh:UPDATE_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:UPDATE_TIMEOUT) Res property description: Maximum execution time allowed for Update method.`
	
	`(oracle1:oracle1-lh:UPDATE_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:UPDATE_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: VALIDATE_TIMEOUT`
	
	`(oracle1:oracle1-lh:VALIDATE_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:VALIDATE_TIMEOUT) Res property description: Maximum execution time allowed for Validate method.`
	
	`(oracle1:oracle1-lh:VALIDATE_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:VALIDATE_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: STOP_TIMEOUT`
	
	`(oracle1:oracle1-lh:STOP_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:STOP_TIMEOUT) Res property description: Maximum execution time allowed for Stop method.`
	
	`(oracle1:oracle1-lh:STOP_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:STOP_TIMEOUT) Res property value: 300`
	
	`(oracle1:oracle1-lh) Res property name: START_TIMEOUT`
	
	`(oracle1:oracle1-lh:START_TIMEOUT) Res property class: standard`
	
	`(oracle1:oracle1-lh:START_TIMEOUT) Res property description: Maximum execution time allowed for Start method.`
	
	`(oracle1:oracle1-lh:START_TIMEOUT) Res property type: int`
	
	`(oracle1:oracle1-lh:START_TIMEOUT) Res property value: 500`
	
	`(oracle1:oracle1-lh) Res property name: CheckNameService`
	
	`(oracle1:oracle1-lh:CheckNameService) Res property class: extension`
	
	`(oracle1:oracle1-lh:CheckNameService) Res property description:`

Command	/usr/cluster/bin/scrgadm -pvv

```
Name service check flag
(oracle1:oracle1-lh:CheckNameService) Res property pernode: False
 (oracle1:oracle1-lh:CheckNameService) Res property type: boolean
 (oracle1:oracle1-lh:CheckNameService) Res property value: TRUE
 (oracle1:oracle1-lh) Res property name: NetIfList
 (oracle1:oracle1-lh:NetIfList) Res property class: extension
 (oracle1:oracle1-lh:NetIfList) Res property description: List of
IPMP groups on each node
(oracle1:oracle1-lh:NetIfList) Res property pernode: False
 (oracle1:oracle1-lh:NetIfList) Res property type: stringarray
 (oracle1:oracle1-lh:NetIfList) Res property value: ipmp1@1
ipmp1@2
 (oracle1:oracle1-lh) Res property name: HostnameList
(oracle1:oracle1-lh:HostnameList) Res property class: extension
 (oracle1:oracle1-lh:HostnameList) Res property description: List
of hostnames this resource manages
 (oracle1:oracle1-lh:HostnameList) Res property pernode: False
 (oracle1:oracle1-lh:HostnameList) Res property type: stringarray
(oracle1:oracle1-lh:HostnameList) Res property value: oracle1
...
```

Command	/usr/cluster/bin/scrgadm -pvv
Values taken	• Groups: ▪ Name ▪ Description ▪ Management state ▪ Mode (failover/scalable) ▪ Maximum primaries ▪ Desired primaries ▪ Nodes list ▪ Is system ▪ Autostart on new cluster ▪ Failback • Resources: ▪ Name ▪ Description ▪ Type ▪ Failover mode ▪ Retry interval ▪ Retry count
Comments	Based on the extracted value, Discovery creates Resource Groups with attributes and Resources with attributes. LogicalHostname handling: for this type of resource Discovery extracts an additional **HostnameList** property that contains the host names that this resource manages. Host names are resolved to IPs. Resolved IPs are attached to the **ClusteredServer** CIT.

Get Cluster Interconnection Information

Command	/usr/cluster/bin/scstat -W

Example of output	`-- Cluster Transport Paths --`
	` Endpoint Endpoint Status`
	` -------- -------- ------`
	` Transport path: node1:bge3 node2:nxge11 Path online`
	` Transport path: node1:nxge3 node2:nxge3 Path online`

Values taken	Output contains the list of transport paths with their statuses. For each path which is online we get source interface on a source node and target interface on a target node.
Comments	Such transport path will be reported with Layer2 links from source interface to target interface. To report the remote interface (located on a node which is not the one connected to), the MAC addresses described below are retrieved.

Command	/usr/cluster/bin/scconf -p

Example of output	`...`
	`Cluster install mode: disabled`
	`Cluster private net: 172.2.0.0`
	`Cluster private netmask: 255.255.255.192`
	`Cluster maximum nodes: 6`
	`Cluster maximum private networks: 4`
	`Cluster new node authentication: unix`
	`Cluster authorized-node list: <. - Exclude all nodes>`
	`Cluster transport heart beat timeout: 10000`
	`Cluster transport heart beat quantum: 1000`
	`Round Robin Load Balancing UDP session timeout: 480`
	`Cluster nodes: node1 node2`
	`Cluster node name: node1`
	` Node ID: 1`
	` Node enabled: yes`
	` Node private hostname: clusternode1-priv`
	` Node quorum vote count: 1`
	` Node reservation key: 0x4A7ADDD300000001`
	` Node zones: <NULL>`
	` CPU shares for global zone: 1`
	` Minimum CPU requested for global zone: 1`
	` Node transport adapters: nxge3 bge3`
	` Node transport adapter: nxge3`
	` Adapter enabled: yes`
	` Adapter transport type: dlpi`
	` Adapter property: device_name=nxge`

Example of output (continued)	
	```
	Adapter property: device_instance=3
	 Adapter property: lazy_free=1
	 Adapter property: dlpi_heartbeat_timeout=10000
	 Adapter property: dlpi_heartbeat_quantum=1000
	 Adapter property: nw_bandwidth=80
	 Adapter property: bandwidth=70
	 Adapter property: ip_address=172.2.0.9
	 Adapter property: netmask=255.255.255.248
	Adapter port names: 0
	 Adapter port: 0
	 Port enabled: yes
	 Node transport adapter: bge3
	 Adapter enabled: yes
	 Adapter transport type: dlpi
	 Adapter property: device_name=bge
	Adapter property: device_instance=3
	 Adapter property: lazy_free=1
	 Adapter property: dlpi_heartbeat_timeout=10000
	 Adapter property: dlpi_heartbeat_quantum=1000
	 Adapter property: nw_bandwidth=80
	 Adapter property: bandwidth=70
	 Adapter property: ip_address=172.2.0.17
	Adapter property: netmask=255.255.255.248
	 Adapter port names: 0
	 Adapter port: 0
	Port enabled: yes
	...
	```

Values taken	Private network address. List of interfaces that are used in cluster interconnect: name and IP address assigned.

Command	/usr/sbin/arp 172.2.0.10
Example of output	`172.2.0.10 (172.2.0.10) at 0:21:a8:39:33:a9`
Values taken	MAC
Comments	Discovery resolves the MAC address of remote interface via arp. If it cannot be resolved, Discovery does not report the transport path as Layer2 link.

Get Quorum Configuration

Command	/usr/cluster/bin/scstat -q

Example of output	`-- Quorum Summary from latest node reconfiguration --`
	` Quorum votes possible: 3`
	` Quorum votes needed: 2`
	` Quorum votes present: 3`
	`-- Quorum Votes by Node (current status) --`
	` Node Name Present Possible Status`
	` --------- ------- -------- ------`
	` Node votes: node1 1 1 Online`
	` Node votes: node2 1 1 Online`
	`-- Quorum Votes by Device (current status) --`
	` Device Name Present Possible Status`
	` ----------- ------- -------- ------`
	` Device votes: clusterquo1 1 1 Online`

Values taken	The quorum status information.
Comments	The details about quorum devices are appended to the Quorum Configuration config file.

Chapter 26: Veritas Discovery

This chapter includes:

Overview

A Veritas Cluster group is a collection of dependent or related resources that is managed as a single unit. Each Veritas Cluster group is linked to a designated node, which is responsible for activating the resources contained in the group. If a failure occurs in the designated node, the responsibility for activating the resources is switched over to a different node.

Veritas Clusters are composed of several clustered servers. Each server is responsible for running certain services and applications. The servers are used as backups for one another. When a system components fails, another server takes over to provide the necessary service.

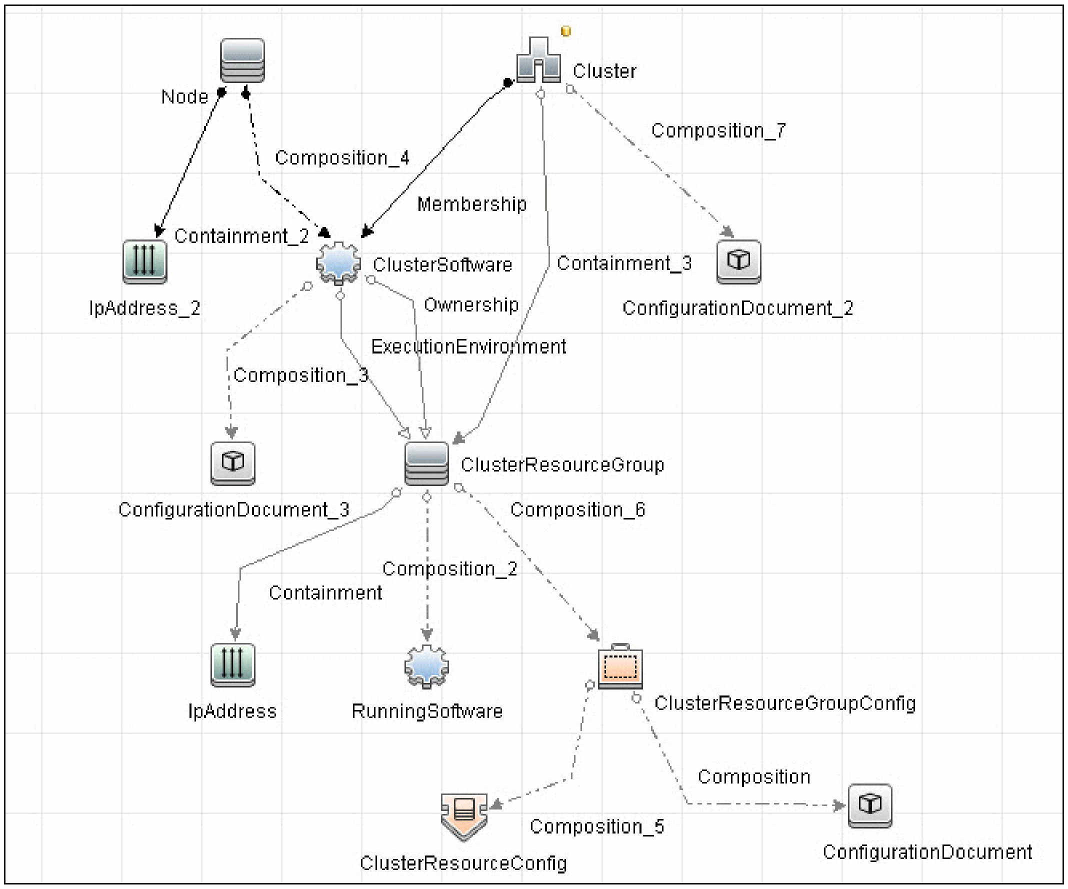
Supported Versions

Veritas Cluster Server (VCS) for UNIX 2.x, 3.x, 4.x, 5.x

Topology

This view shows the top layer of the Veritas Cluster topology. It displays the discovered Veritas Cluster and the clustered software resources that are members of that cluster. Each software resource is linked by a **membership** relationship to the Veritas Cluster.

Note: For a list of discovered CITs, see "Veritas Cluster by Shell Job" on the next page.



How to Discover Veritas Cluster Servers

The Veritas Cluster discovery process enables you to discover Veritas Cluster Servers (VCS), and their member machines (also referred to as nodes), that activate the discovered resources provided by the cluster.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the SSH/Telnet protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Run the following jobs in the following order:

   a. Run the **Host Connection by Shell** job.

   b. Run the **Host Applications by Shell** job.

   c. Run the **Veritas Cluster by Shell** job. For job details, see "Veritas Cluster by Shell Job" below.

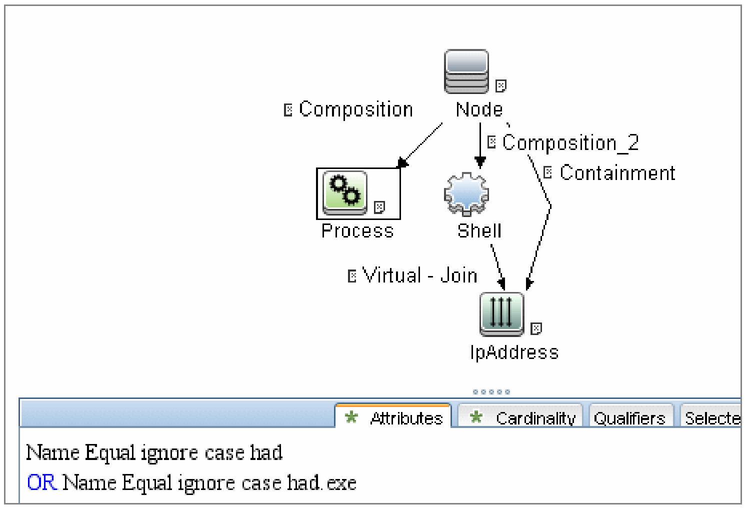
Veritas Cluster by Shell Job

This section includes:

- "Veritas Cluster by Shell Job" above
- "Veritas Cluster by Shell Job" above
- "Veritas Cluster by Shell Job" above
- "Veritas Cluster by Shell Job" above

Trigger Query

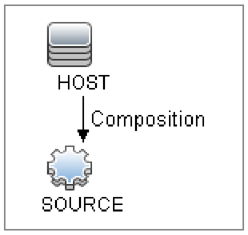
- **Trigger query**:



Name Equal ignore case had
OR Name Equal ignore case had.exe

Adapter

- **Input query**:



Used Scripts

- file_ver_lib.py
- Veritas_Cluster_Topology.py

Discovered CITs

- ClusterSoftware
- Composition
- ConfigurationDocument
- Containment
- Dependency
- IpAddress
- IpServiceEndpoint
- Membership
- Node
- Ownership
- RunningSoftware
- Usage
- VCS Resource Group
- VCS resource
- Veritas Cluster

Note: To view the topology, see "Topology" on page 363.

Part 5: Clustering and Load Balancing > Load Balancers

Chapter 27: Load Balancer Discovery

This chapter includes:

Overview

DFM discovers the following load balancers:

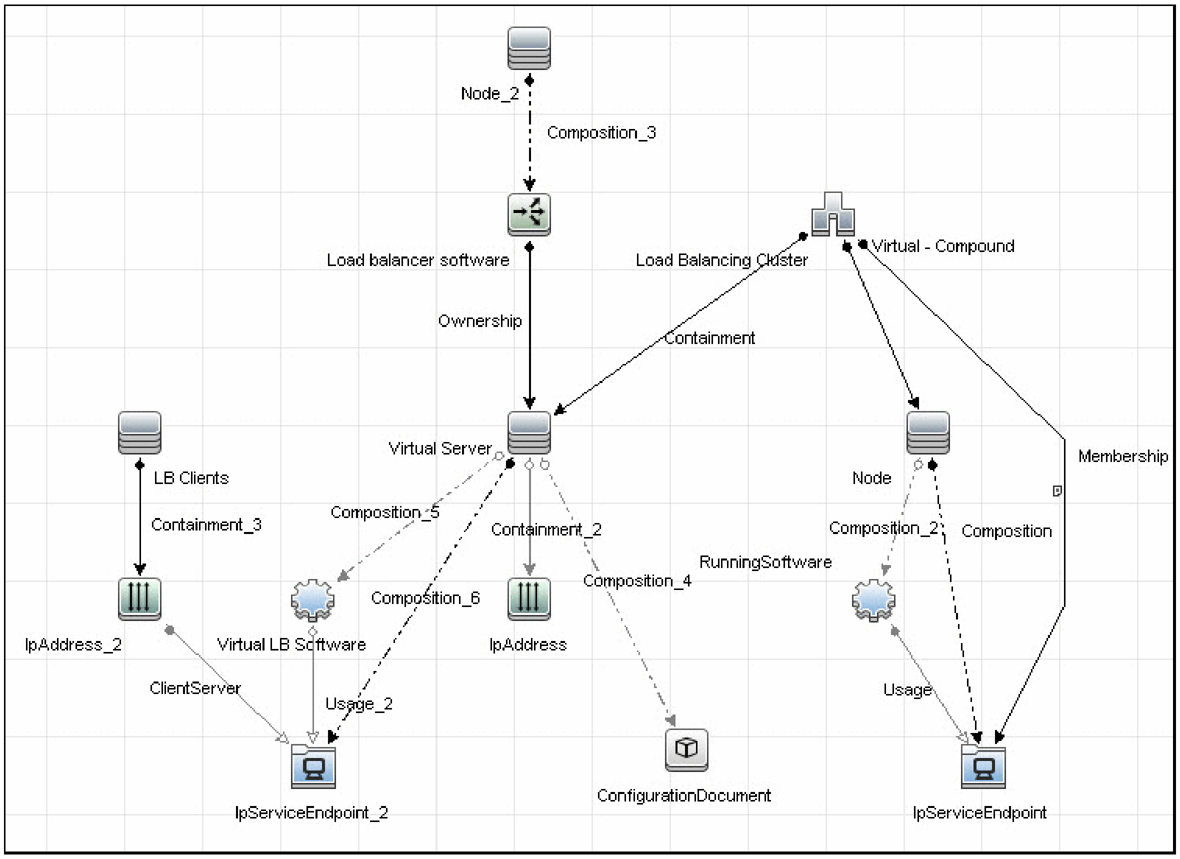
- F5 BIG-IP Local Traffic Manager (LTM)
- Nortel Application Switches (formerly known as Alteon Application Switches)
- Cisco Content Services Switches (CSS)

Supported Versions

The supported version for each load balancer is as follows:

- **F5 BIG-IP Local Traffic Manager:** versions 4, 9 and 10.
- **Nortel Application Switches:** no known limitations.
- **Cisco Content Services Switches:** no known limitations.

Topology



Note: For a list of discovered CITs, see "Discovered CITs" on page 385.

How to Discover Load Balancers

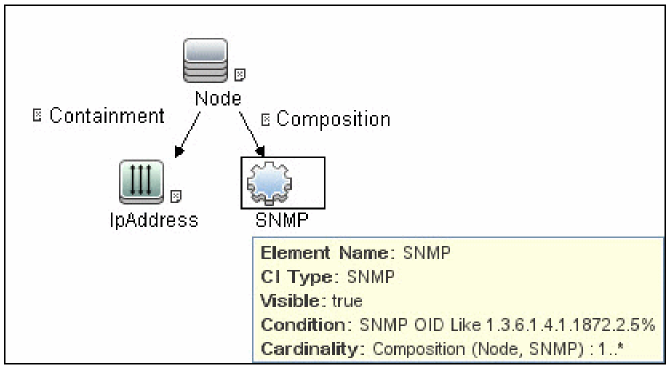
This task explains how to discover load balancers and includes the following steps:

- "Prerequisites" below
- "Run the discovery" on page 372

1. **Prerequisites**

   Run the **Host Connection by SNMP** job to discover and create SNMP CIs which answer the following requirements:

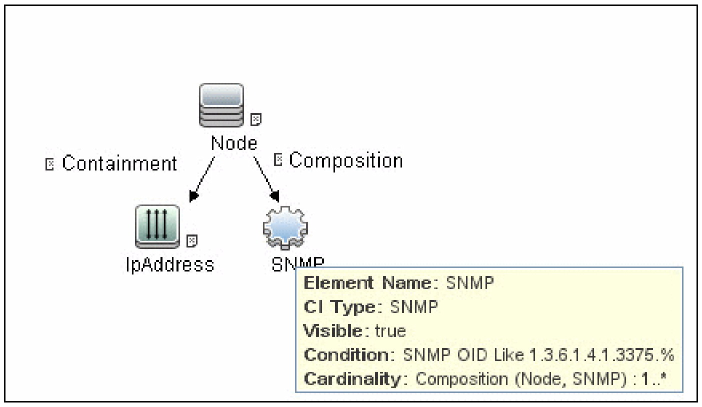
- To be the trigger query for the **Alteon application switch by SNMP** job with the following condition:

  Element Name: SNMP
  CI Type: SNMP
  Visible: true
  Condition: SNMP OID Like 1.3.6.1.4.1.1872.2.5%
  Cardinality: Composition (Node, SNMP) : 1..*

  ```
  SNMP OID Like 1.3.6.1.4.1.1872.2.5%
  ```

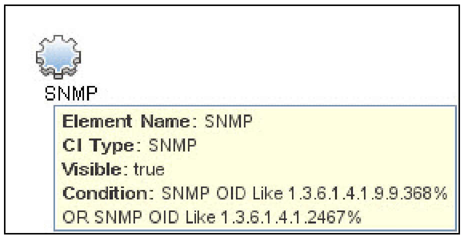
- To be the trigger query for the **F5 BIG-IP LTM by SNMP** job with the following condition:

  Element Name: SNMP
  CI Type: SNMP
  Visible: true
  Condition: SNMP OID Like 1.3.6.1.4.1.3375.%
  Cardinality: Composition (Node, SNMP) : 1..*

  ```
  SNMP OID Like 1.3.6.1.4.1.3375%
  ```

- To be the trigger query for the **Cisco CSS by SNMP** job with the following condition:



Element Name: SNMP
CI Type: SNMP
Visible: true
Condition: SNMP OID Like 1.3.6.1.4.1.9.9.368% OR SNMP OID Like 1.3.6.1.4.1.2467%

```
SNMP OID Like 1.3.6.1.4.1.9.9.368% OR 1.3.6.1.4.1.2467%
```

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   - **Host Connection by SNMP.** For details on the prerequisites to running a load balancer job, see "Prerequisites" on page 370.
   - Run any of the following jobs:
     - **F5 BIG-IP LTM by SNMP**
     - **Alteon application switch by SNMP**
     - **Cisco CSS by SNMP**

     For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Alteon Application Switch by SNMP Job

This job discovers Nortel (Alteon) load balancers and all related CIs.

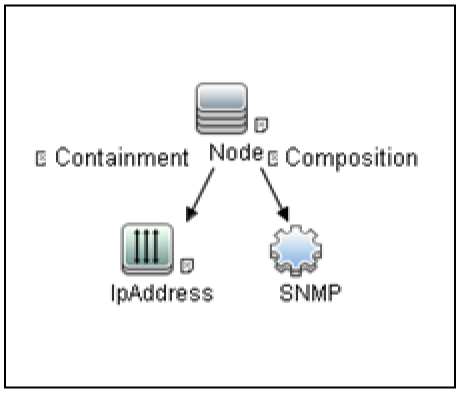
Queried SNMP Tables

The following SNMP tables are queried:

Table Name	Name From MIB	OID
Virtual servers	slbCurCfgVirtServer Table	1.3.6.1.4.1.1872.2.5.4.1.1.4.2.1

Table Name	Name From MIB	OID
Virtual services	slbCurCfgVirtServices Table	1.3.6.1.4.1.1872.2.5.4.1.1.4.5.1
Real groups	slbCurCfgGroupEntry	1.3.6.1.4.1.1872.2.5.4.1.1.3.3.1
Real servers	slbCurCfgRealServer Table	1.3.6.1.4.1.1872.2.5.4.1.1.2.2.1
Port links	slbCurCfgRealServPortTable	1.3.6.1.4.1.1872.2.5.4.1.1.2.5.1
Ports	slbCurCfgPortTable	1.3.6.1.4.1.1872.2.5.4.1.1.5.2.1

Trigger Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
SNMP	SNMP OID Like 1.3.6.1.4.1.1872.2.5%
Node	none

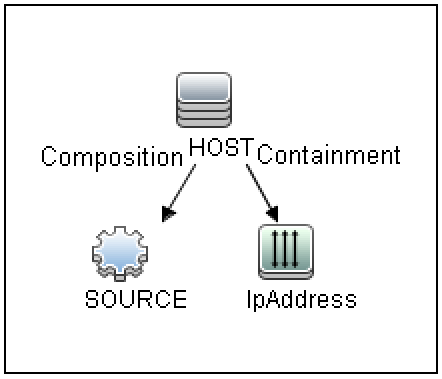
Adapter Information

This job uses the **Alteon_app_switch_by_SNMP** adapter.

Input CIT

SNMP

Input TQL Query



Node Name	Condition
SOURCE	None
IpAddress	IpAddressType Equal IPv4
HOST	None

Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
oid	${SOURCE.snmp_oid}

Used Scripts

- snmputils.py
- Alteon_app_switch_by_SNMP.py

Discovered CITs

- Alteon application switch
- Composition

- ConfigurationDocument
- Containment
- IpAddress
- IpServiceEndpoint
- Load Balancer
- Load Balancing Cluster
- Membership
- Node
- Ownership

F5 BIG-IP LTM by SNMP Job

This job discovers the F5 BIG-IP Local Traffic Manager (LTM) by SNMP. DFM chooses all SNMPs related to F5 and runs against them.

The package supports F5 BIG-IP LTM, versions 11,10, 9 and 4.

Queried SNMP Tables

The following SNMP tables are queried for versions 10 and 9:

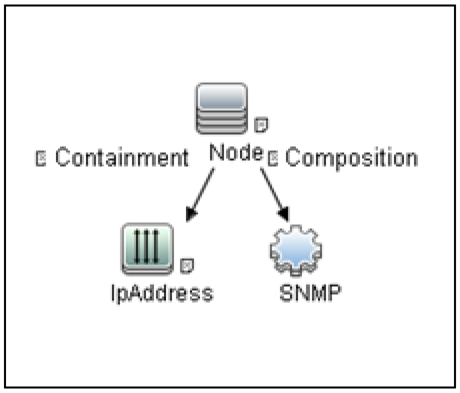
Table Name	Name From MIB	OID
General information	sysProduct	1.3.6.1.4.1.3375.2.1.4
Virtual servers	ltmVirtualServTable	1.3.6.1.4.1.3375.2.2.10.1.2.1
Pools	ltmPoolTable	1.3.6.1.4.1.3375.2.2.5.1.2.1
Pools to server	ltmVirtualServPool Table	1.3.6.1.4.1.3375.2.2.10.6.2.1
Pool members	ltmPoolMemberTable	1.3.6.1.4.1.3375.2.2.5.3.2.1

Table Name	Name From MIB	OID
Rules to servers	ltmVirtualServRule Table	1.3.6.1.4.1.3375.2.2.10.8.2.1
Rules	ltmRuleTable	1.3.6.1.4.1.3375.2.2.8.1.2.1

The following SNMP tables are queried for version 4:

Table Name	Name From MIB	OID
General information	globalAttributes	1.3.6.1.4.1.3375.1.1.1.1
Virtual servers	virtualServerTable	1.3.6.1.4.1.3375.1.1.3.2.1
Pools	poolTable	1.3.6.1.4.1.3375.1.1.7.2.1
Pool members	poolMemberTable	1.3.6.1.4.1.3375.1.1.8.2.1

Trigger Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
SNMP	SNMP OID Like 1.3.6.1.4.1.3375.%
Node	None

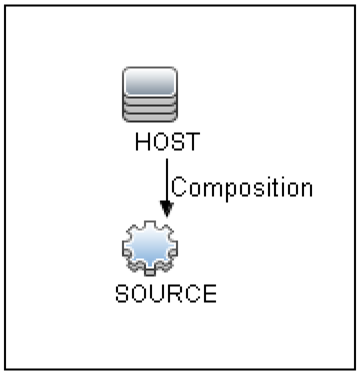
Adapter Information

This job uses the **F5_BIGIP_LTM_by_SNMP** adapter.

Input CIT

SNMP

Input TQL Query



Node Name	Condition
SOURCE	None
HOST	None

Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Used Scripts

- snmputils.py
- F5_BIGIP_LTM_by_SNMP.py

Discovered CITs

- Composition
- ConfigurationDocument

- Containment
- F5 BIG-IP LTM
- IpAddress
- IpServiceEndpoint
- Load Balancer
- Load Balancing Cluster
- Membership
- Node
- Ownership

F5 BIG-IP LTM by Shell Job

This job discovers the F5 BIG-IP Local Traffic Manager (LTM) by Shell. DFM chooses all shells related to F5 and runs against them.

Versions

The package supports F5 BIG-IP LTM, versions 10.x and 11.x.

Prerequisites

This adapter needs SSH protocol credentials which include username and password.

Note: Since the F5 configuration files are readable for all users and writable for root, the user is not required to login as root.

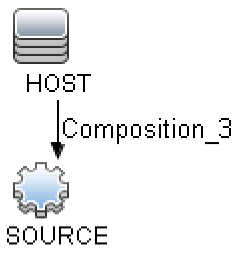
Adapter Information

This job uses the **F5_BIGIP_LTM_by_Shell** adapter. The adapter parses and fetches data from the configuration document of F5 BIG-IP LTM by using Shell command.

Input CIT

Shell

Input TQL Query



Node Name	Condition
SOURCE	None
HOST	None

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}

Used Scripts

- F5_BIGIP_LTM_by_Shell.py

Discovered CITs

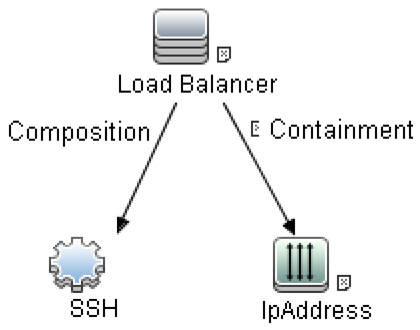
- Composition
- ConfigurationDocument
- Containment
- F5 BIG-IP LTM
- IpAddress
- IpServiceEndpoint
- Load Balancer
- Load Balancing Cluster
- Membership
- Node
- Ownership

Required Permissions

Permission	Operation	Usage Description	Objects and Parameters
Shell	exec	Basic Login	UNIX: date +%z UNIX: echo $? UNIX: echo $SHELL UNIX: locale -a UNIX: uname -a UNIX: uname -r

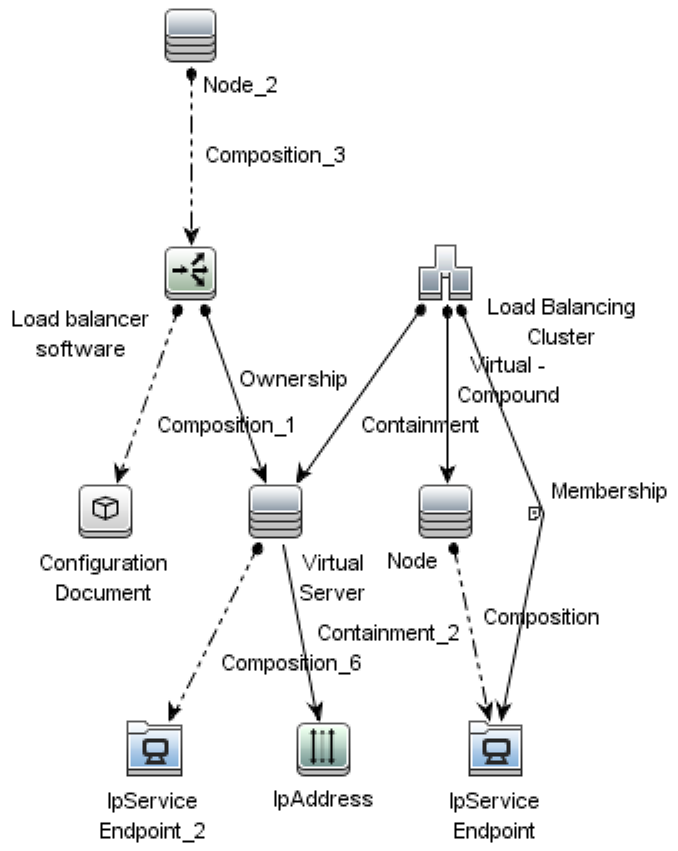
Permission	Operation	Usage Description	Objects and Parameters
Shell	exec	Discover files and F5 details	UNIX: cat <file_path> UNIX: find <folder_path> UNIX: ls -lA <folder_path> UNIX: perl -e UNIX: ps -eo user,pid, lstart,command --cols 2530 --no-headers UNIX: readlink

Trigger TQL



Node Name	Condition
Load Balancer	None
SSH	None
IpAddress	NOT IP Probe Name Is null

Topology



Cisco CSS by SNMP Job

This job discovers Cisco Content Services Switches by SNMP. It supports all versions of Cisco CSS.

To run this discovery, activate the **Cisco CSS by SNMP** job. DFM chooses all SNMPs related to Cisco CSS and runs against them.

Note: Some services may not be discovered by this package if no content rule is defined for them.

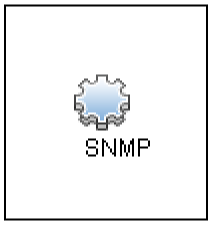
Discovery of CSS is based on three tables: **apCntTable**, **apSvcTable**, and **apCntsvcTable** (see the following table):

- **apCntTable** provides information about virtual addresses, virtual services, and pools.
- **apSvcTable** provides information about physical hosts included in the pool.
- **apCntsvcTable** describes which host is included in which pool.

apSvcTable can contain entries for which there is no corresponding row in **apCntsvcTable**. In this case, such hosts are skipped.

Table Name	Name from MIB	OID
CNT	apCntTable	1.3.6.1.4.1.2467.1.16.4.1 or 1.3.6.1.4.1.9.9.3681.16.4.1
SVC	apSvcTable	1.3.6.1.4.1.2467.1.15.2.1 or 1.3.6.1.4.1.9.9.3681.15.2.1
CNT to SVC	apCntsvcEntry	1.3.6.1.4.1.2467.1.18.2.1 or 1.3.6.1.4.1.9.9.3681.18.2.1

Trigger Query



Node Name	Condition
SNMP	SNMP OID Like 1.3.6.1.4.1.9.9.368% OR SNMP OID Like 1.3.6.1.4.1.2467%

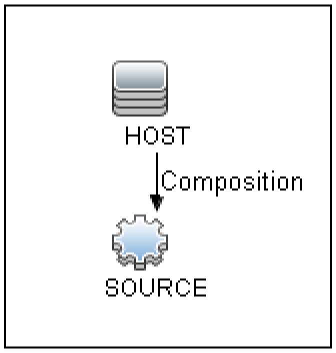
Adapter Information

This job uses the **Cisco CSS by SNMP** adapter.

Input CIT

SNMP

Input TQL Query



Node Name	Condition
SOURCE	None
HOST	None

Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
hostId	${HOST.root_id}

Used Scripts

- snmputils.py
- Cisco_CSS_by_SNMP.py

Discovered CITs

- Cisco CSS
- Composition
- Containment

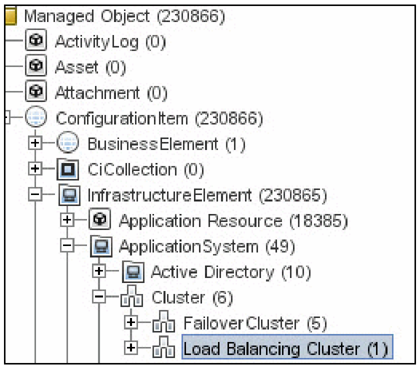
- IpAddress
- IpServiceEndpoint
- Load Balancer
- Load Balancing Cluster
- Membership
- Node
- Ownership

Discovered CITs

The following CITs model load balancer topology:

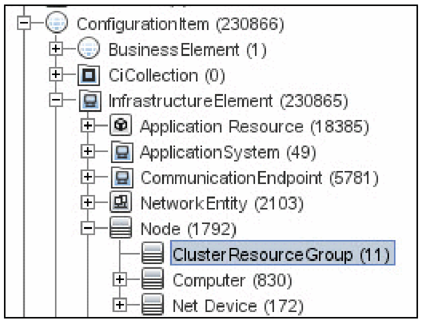
- **Load Balancer Software**

  This CIT represents software that provides load balancing solutions. For details on the supported load balancers, see "Overview" on page 369.



- **Clustered Server**

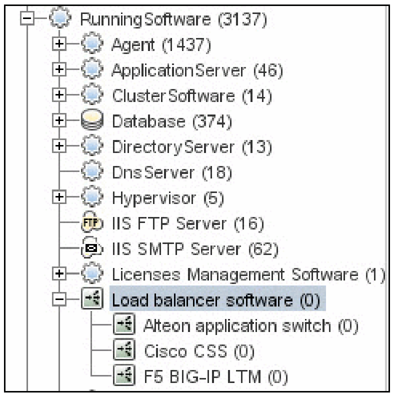
  A clustered server is a traffic-management object on the system that can balance traffic load across a pool of servers. Clustered servers increase the availability of resources for processing client requests. The primary function of a clustered server is to receive requests and distribute them to pool members according to criteria you specify.



```
ConfigurationItem (230866)
    BusinessElement (1)
    CiCollection (0)
    InfrastructureElement (230865)
        Application Resource (18385)
        ApplicationSystem (49)
        CommunicationEndpoint (5781)
        NetworkEntity (2103)
        Node (1792)
            ClusterResourceGroup (11)
            Computer (830)
            Net Device (172)
```

- **Load Balancing Cluster**

  A load balancing cluster (or pool) is a logical set of devices that are grouped together to receive and process traffic. Instead of sending client traffic to the destination IP address specified in the client request, the virtual server sends the request to any of the servers that are members of that pool. This helps to efficiently distribute the load on your server resources.



```
RunningSoftware (3137)
    Agent (1437)
    ApplicationServer (46)
    ClusterSoftware (14)
    Database (374)
    DirectoryServer (13)
    DnsServer (18)
    Hypervisor (5)
    IIS FTP Server (16)
    IIS SMTP Server (62)
    Licenses Management Software (1)
    Load balancer software (0)
        Alteon application switch (0)
        Cisco CSS (0)
        F5 BIG-IP LTM (0)
```

Note: To view the topology, see "Topology" on page 370.

Chapter 28: Microsoft Network Load Balancing (NLB) Discovery

This chapter includes:

Overview

Network Load Balancing (NLB) distributes IP traffic to multiple copies (or instances) of a TCP/IP service, such as a Web server, each running on a host within the cluster. NLB transparently partitions the client requests among the hosts and lets the clients access the cluster using one or more virtual IP addresses. From the client's point of view, the cluster appears to be a single server that answers these client requests. Each server receives all client requests, but NLB decides which server should respond.

All components responsible for the Microsoft NLB cluster are bundled in the **Microsoft_NLB_Cluster.zip** package.

To discover MS-NLB, see "How to Discover Microsoft Network Load Balancing Systems" on page 390.

See also:

- "Components of the Network Load Balancing Architecture" on page 398
- "Glossary" on page 400

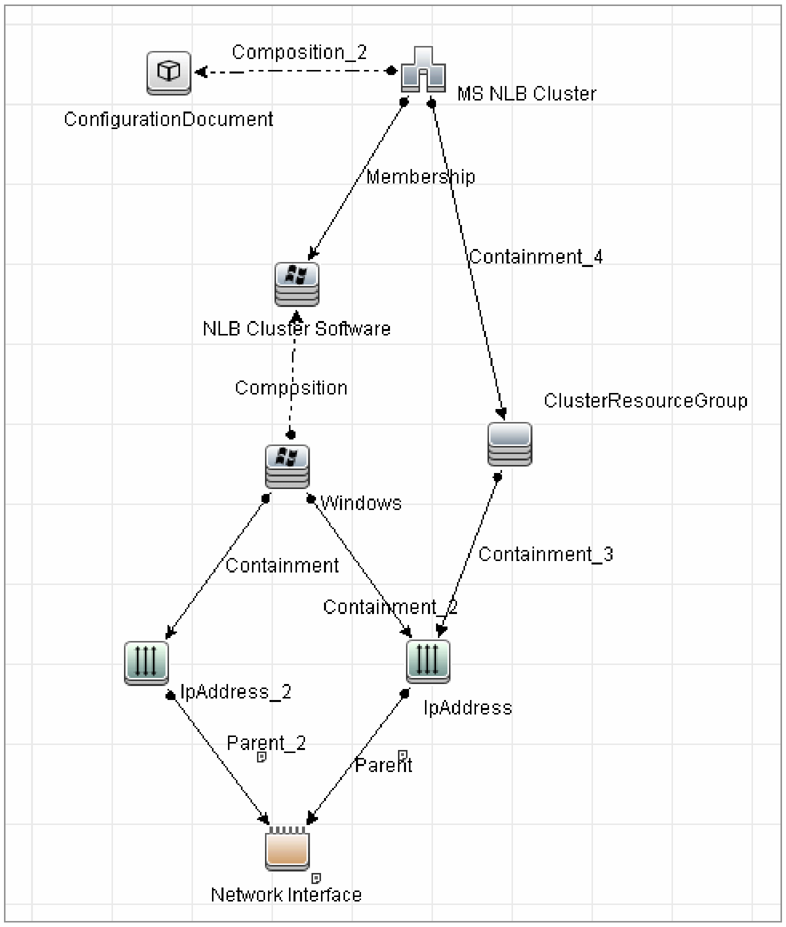
Supported Versions

This discovery supports Microsoft Network Load Balancer versions 2000, 2003, 2008.

Topology

The following image displays the topology of the MS NLB discovery:

Note: For a list of discovered CITs, see "MS NLB by NTCMD or UDA Job" on page 392.



How to Discover Microsoft Network Load Balancing Systems

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the NTCMD protocol, or Universal Discovery protocol if UD Agent installed on the Exchange server.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Verify that the user defined in the NTCMD protocol is granted administration rights for Shell execution on the remote machine.

   The NTCMD protocol retrieves information about NLB by executing the **wlbs params** command.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Activate the following jobs in the following order:

   - The **Host Connection by Shell** job to discover Windows machines that act as the triggers for the NLB discovery.
   - The **MS NLB by NTCMD or UDA** job to connect to the host by NTCMD and retrieve the MS NLB Cluster topology. For job details, see "MS NLB by NTCMD or UDA Job" on page 392.

   For details on the discovery mechanism, see "MS NLB by NTCMD or UDA Job" on page 392.

How to Discover NLB Using Command Line Utility

You can discover NLB by running the **nlb.exe** command line utility.

This utility runs with the **params** key and outputs information about all NLB clusters on a discovered machine.

- If NLB is not installed on a Windows 2003 Server machine, the output is as follows:

```
WLBS Cluster Control Utility V2.4 (c) 1997-2003 Microsoft Corporation.
WLBS is not installed on this system or you do not have sufficient privileges to
administer the cluster.
```

- If an NLB cluster is set up on the machine, the output is as follows:

```
Cluster 192.168.0.222
Retrieving parameters
Current time              = 9/3/2009 1:02:38 PM
HostName                  = ddmvm-2k3-s
ParametersVersion         = 4
CurrentVersion            = 00000204
EffectiveVersion          = 00000201
InstallDate               = 4A9E51F5
HostPriority              = 1
ClusterIPAddress          = 192.168.0.222
ClusterNetworkMask        = 255.255.255.0
DedicatedIPAddress        = 192.168.0.2
DedicatedNetworkMask      = 255.255.255.0
McastIPAddress            = 0.0.0.0
ClusterName               = cluster2.domain.com
ClusterNetworkAddress     = 03-bf-c0-a8-00-de
IPToMACEnable             = ENABLED
MulticastSupportEnable    = ENABLED
IGMPSupport               = DISABLED
MulticastARPEnable        = ENABLED
MaskSourceMAC             = ENABLED
AliveMsgPeriod            = 1000
AliveMsgTolerance         = 5
NumActions                = 100
NumPackets                = 200
NumAliveMsgs              = 66
DescriptorsPerAlloc       = 512
MaxDescriptorAllocs       = 512
TCPConnectionTimeout      = 60
IPSecConnectionTimeout    = 86400
```

```
FilterICMP                 = DISABLED
ClusterModeOnStart         = STARTED
HostState                  = STARTED
PersistedStates            = NONE
ScaleSingleClient          = DISABLED
NBTSupportEnable           = ENABLED
NetmonAliveMsgs            = DISABLED
IPChangeDelay              = 60000
ConnectionCleanupDelay     = 300000
RemoteControlEnabled       = DISABLED
RemoteControlUDPPort       = 2504
RemoteControlCode          = 00000000
RemoteMaintenanceEnabled   = 00000000
BDATeaming                 = NO
TeamID                     =
Master                     = NO
ReverseHash                = NO
IdentityHeartbeatPeriod    = 10000
IdentityHeartbeatEnabled   = ENABLED
PortRules (1):
      VIP       Start  End  Prot   Mode    Pri Load Affinity
--------------- ----- ----- ---- -------- --- ---- --------
All                 0 65535 Both Multiple      Eql Single
```

No special rules are used for mapping the output to the CITs; all CI attributes repeat the output data names. Data is verified by comparing it to cluster nodes that have already been discovered.

MS NLB by NTCMD or UDA Job

This section includes:

- "MS NLB by NTCMD or UDA Job" above
- "MS NLB by NTCMD or UDA Job" above
- "MS NLB by NTCMD or UDA Job" above
- "MS NLB by NTCMD or UDA Job" above
- "MS NLB by NTCMD or UDA Job" above
- "MS NLB by NTCMD or UDA Job" above

Discovery Mechanism

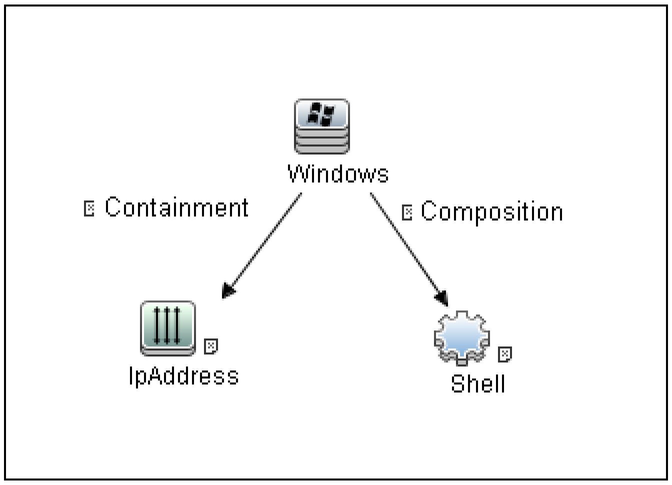
DFM triggers on Windows machines with more than one (two or more) IP addresses, and collects information using the **nlb.exe** command line utility. (In earlier versions of the Windows 2000 family, **wlbs.exe** is used.) These utilities enable the retrieval of all NLB-related information. For details, see "MS NLB by NTCMD Adapter" on the next page.

There is no need for DFM to collect information from every participating node to verify that an MS NLB cluster system exists: even one single machine running the software is considered a cluster machine. If more machines are discovered that include the NLB service (with the same settings as the first machine), the NLB cluster begins the convergence process.

Furthermore, cluster information is collected by discovering one node at a time because nodes participating in a cluster do not include information about the other participants.

Trigger Query

- Trigger CIT: **NTCMD**
- Trigger query:



- **CI Attribute Condition:** NTCMD or UDA running on a Windows machine with at least two IP addresses.

Name	Category	Description
ntcmd_with_2_IP	Trigger	Used by the **MS NLB by NTCMD or UDA** job
MS NLB topology	View	Used by the **MS NLB Topology** view

Adapter

This job uses the **MS NLB by NTCMD** adapter. For details, see "MS NLB by NTCMD Adapter" below.

Views

- Microsoft NLB topology

Used Scripts

- ms_nlb_report_utils.py
- ms_nlb_ntcmd.py

Discovered CITs

- Composition
- ConfigurationDocument. For details, see "MS NLB by NTCMD Adapter" below.
- Containment
- IpAddress
- Membership
- MS NLB Cluster. For details, see "MS NLB by NTCMD Adapter" below.
- NLB Cluster Software. For details, see "MS NLB by NTCMD Adapter" below.
- Node

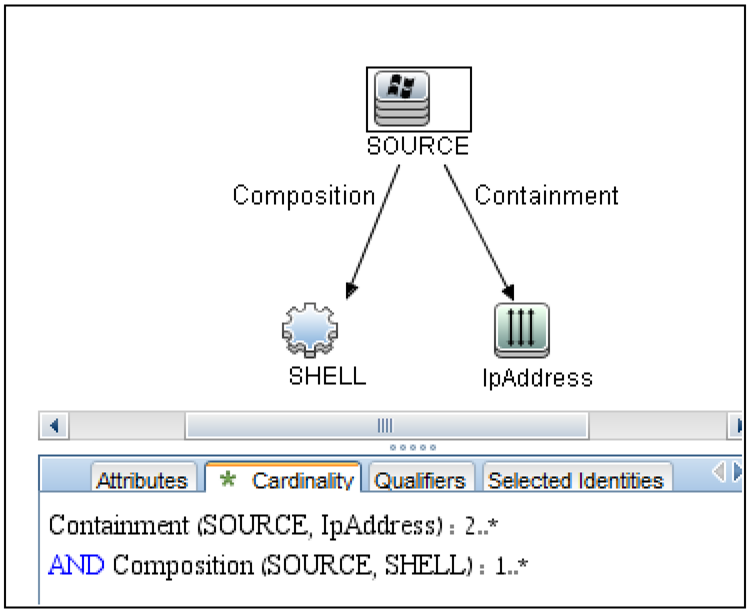
Note: To view the topology, see "Topology" on page 389.

MS NLB by NTCMD Adapter

This section contains details about the adapter.

Input Query

NTCMD or UDA running on a Windows machine with at least two IP addresses:



Containment (SOURCE, IpAddress) : 2..*
AND Composition (SOURCE, SHELL) : 1..*

Triggered CI Data

Name	Value
credentialsId	${NTCMD.credentials_id}
ip_address	${IpAddress.name}

MS NLB Cluster CIT

The CIT represents information regarding the NLB cluster.

- CIT name: **ms_nlb_cluster**
- Parent CIT name: **loadbalancecluster**
- Relationships

Start Node	Start Node Cardinality	Name	End Node	End Node Cardinality
ms_nlb_ cluster	1..*	membership	nlb_ clustersoftware	1..*

The Cluster IP address is a key field, as this is the most reliable way of discovering NLB. By comparison, discovering NLB through the Cluster network address is less reliable as it is dependent on the IP address and the operating mode—Unicast, Multicast, or IGMP. The Cluster domain name is retrieved for the Cluster name.

- Attributes

  The following attributes are specific to the MS NLB Cluster CIT:

Key	Display Name	Attribute Name	Type
X	ClusterIPAddress	cluster_ip_address	String(15)
	ClusterNetworkMask	cluster_network_mask	String(15)
	McastIPAddress	mcast_ip_address	String(15)
	ClusterDomainName	cluster_domain_name	String(256)
	ClusterNetworkAddress	cluster_network_address	MAC Address
	IPToMACEnable	ip_to_mac_enable	Boolean
	MulticastSupportEnable	multicast_support_enable	Boolean
	IGMPSupport	igmp_support	Boolean
	RemoteControlEnabled	remote_control_enabled	Boolean
X	Name	name	String (modified for this CIT)

NLB Cluster Software CIT

The CIT represents information regarding a single machine configuration that is part of an NLB cluster.

- CIT name: **nlb_clustersoftware**
- Parent CIT name: **failoverclustersoftware**

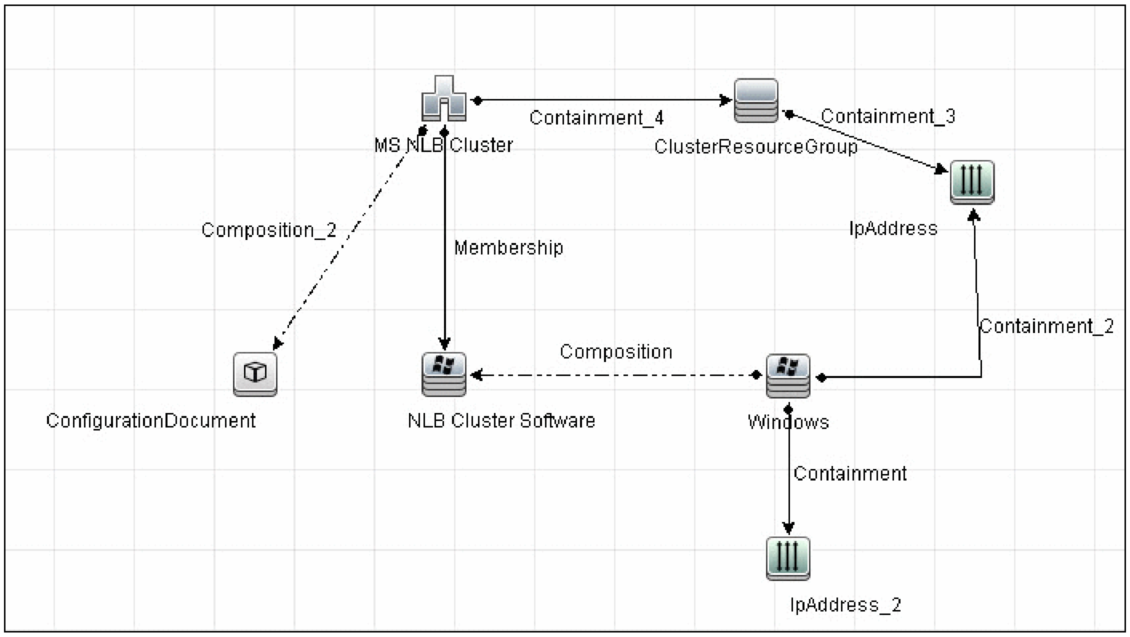
- Relationships

Start Node	Start Node Cardinality	Name	End Node	End Node Cardinality
ms_nlb_ cluster	1..*	membership	nlb_ clustersoftware	1..*
nt	1..*	composition	nlb_ clustersoftware	1..*

- Attributes

Key	Display Name	Type
	ClusterIPAddress	String(15)
	HostPriority	int (1-32)
	ClusterModeOnStart	Started, Suspended, Stopped
	Name	String (NLB Cluster SW)
	Composition	String (32)

ConfigurationDocument (NLB Port Rule)



This CIT retrieves information about each port rule defined for NLB clusters.

Since the Port Rule entity cannot clearly define key attributes, the port rules properties are stored in the properties file (key=value pairs) as follows:

```
portRule1.ServingIP=All
portRule1.StartPort=0
portRule1.EndPort=100
portRule1.Protocol=Both
portRule1.FilteringMode=Multiple
portRule1.Affinity=Single
portRule1.LoadWeight=40
```

Relationships

Start Node	Start Node Cardinality	Name	End Node	End Node Cardinality
nt	1..*	composition	nlb_ clustersoftware	1..*
ms_nlb_ cluster	1..*	membership	nlb_ clustersoftware	1..*

Components of the Network Load Balancing Architecture

Component	Description
Nlb.exe	The Network Load Balancing control program. You use Nlb.exe from the command line to start, stop, and administer Network Load Balancing, as well as to enable and disable ports and to query cluster status.
Nlbmgr.exe	The Network Load Balancing Manager control program. Use this command to start Network Load Balancing Manager.
Wlbs.exe	The former Network Load Balancing control program. This has been replaced by **Nlb.exe**. However, you can still use **Wlbs.exe** rather than **Nlb.exe** if necessary, for example, if you have existing scripts that reference **Wlbs.exe**.
Wlbsprov.dll	The Network Load Balancing WMI provider.
Nlbmprov.dll	The Network Load Balancing Manager WMI provider.
Wlbsctrl.dll	The Network Load Balancing API DLL.

Component	Description
Wlbs.sys	The Network Load Balancing device driver. **Wlbs.sys** is loaded onto each host in the cluster and includes the statistical mapping algorithm that the cluster hosts collectively use to determine which host handles each incoming request.

Glossary

Cluster

A group of independent computers that work together to run a common set of applications and provide the image of a single system to the client and application. The computers are physically connected by cables and programmatically connected by cluster software. These connections allow computers to use problem-solving features such as failover in Server clusters and load balancing in Network Load Balancing (NLB) clusters. For details, refer to http://technet.microsoft.com/en-us/library/cc784941(WS.10).aspx.

Dedicated IP Address

The IP address of a NLB host used for network traffic that is not associated with the NLB cluster (for example, Telnet access to a specific host within the cluster). This IP address is used to individually address each host in the cluster and therefore is unique for each host.

NLB Node

Machine-participant of an NLB cluster. For details, refer to http://technet.microsoft.com/en-us/library/cc758834(WS.10).aspx.

Operating Mode

The NLB cluster has two operating modes:

- In its default unicast mode of operation, NLB reassigns the station (MAC) address of the network adapter for which it is enabled and all cluster hosts are assigned the same MAC (media access control) address.
- In multicast mode, NLB assigns a layer 2 multicast address to the cluster adapter instead of changing the adapter's station address. For details, refer to http://technet.microsoft.com/en-us/library/cc783135(WS.10).aspx.

Port Rules

The NLB driver uses port rules that describe which traffic to load-balance and which traffic to ignore. By default, the NLB driver configures all ports for load balancing. You can modify the configuration of the NLB driver that determines how incoming network traffic is load-balanced on a per-port basis by creating port rules for each group of ports or individual ports as required. Each port rule configures load balancing for client requests that use the port or ports covered by the port range parameter. How you load-balance your applications is mostly defined by how you add or modify port rules, which you create on each host for any particular port range.

Virtual IP Address

An IP address that is shared among the hosts of a NLB cluster. A NLB cluster may also use multiple virtual IP addresses, for example, in a cluster of multihomed Web servers. For details, refer to http://technet.microsoft.com/en-us/library/cc756878(WS.10).aspx.

Part 6: Database

Chapter 29: Database Connections by Host Credentials Discovery

This chapter includes:

Overview

The purpose of this package is to enable database auto-discovery using host level credentials in HP Universal CMDB (UCMDB). In certain cases, a DFM user or administrator does not have detailed information about the database, such as its name or SID, listener port number, and so on. The solution in this package discovers this information with minimal inputs, and enables end-to-end discovery of databases.

DFM extracts database information from various sources, for example, from running process names, Windows service names, the Windows registry, and configuration files, on the database server and build CIs. Discovered Database CIs can be used as triggers for the Database Connection by SQL jobs (for example, the **Oracle Database Connection by SQL** job), to populate database credentials, thus enabling deep discovery using out-of-the-box database topology discovery jobs.

DFM triggers for jobs in this package are set up so that these jobs are seamlessly included in the UCMDB spiral discovery schedule.

The **DB Connections by Shell** and **DB Connections by WMI** jobs in this package use a Shell (NTCMD/SSH/Telnet) or agent (WMI) CI as a trigger, to search for database signatures on a host. These jobs create database CIs with available information, such as instance name or SID and the listener port of the database server. Since database credentials are not used, the username and credentials ID attributes of these CIs are empty.

For more details about these jobs, see:

- "DB Connection by Shell Job" on page 406
- "DB Connection by WMI Job" on page 410

Supported Versions

Database Connections by Host Credentials Discovery supports the following database servers:

- Oracle 9i, 10g, 11g
- Microsoft SQL Server 2000, 2005, 2008, 2008 R2, 2012, and 2014

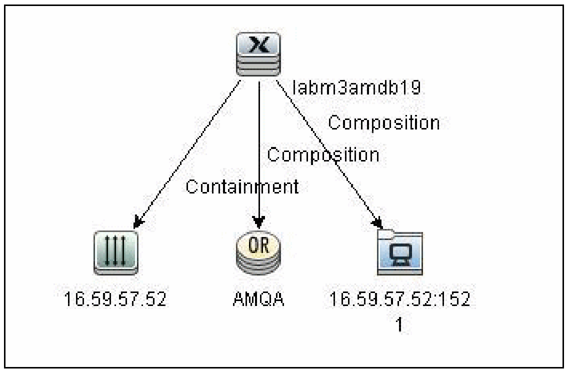
Note: For CP 13.0 and later versions, discovery of DB2 database servers is not supported.

Topology

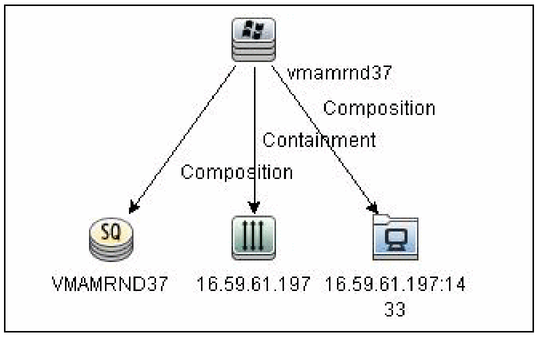
The following images display the topology of the Database Connections by Host Credentials discovery with sample output:

Note: For a list of discovered CITs, see "DB Connection by Shell Job" on the next page and "DB Connection by WMI Job" on page 410.

Oracle



Microsoft SQL



How to Discover Database Connections by Host Credentials

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the following protocols:

   - WMI protocol
   - NTCMD protocol
   - SSH protocol
   - Telnet protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Discover Host Credentials**

   a. Run the **Range IPs by ICMP** job.

   b. Run the **Host Connection by Shell** job.

   c. Run the **Host Connection by WMI** job.

   d. Run the **DB Connections by Shell** job. For details, see "DB Connection by Shell Job" below.

   e. Run the **DB Connections by WMI** job. For details, see "DB Connection by WMI Job" on page 410.

DB Connection by Shell Job

This section includes:

- "DB Connection by Shell Job" above
- "DB Connection by Shell Job" above
- "DB Connection by Shell Job" above
- "DB Connection by Shell Job" above

Discovery Mechanism

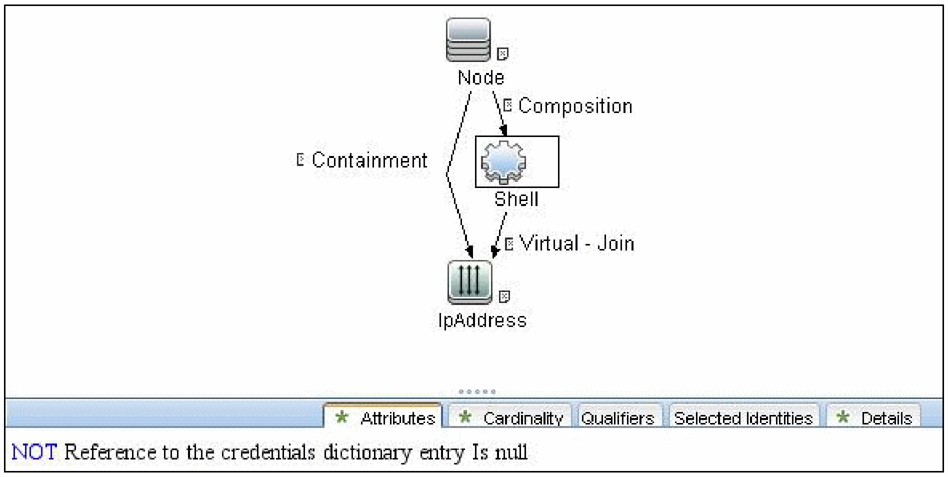
This discovery job attempts to identify configured databases on a host using a Shell client (NTCMD/SSH/Telnet). Once connected, the job creates a list of running processes and server ports associated with each process. On Microsoft Windows operating systems, this job adds a list of installed Windows services to the list.

The job then looks for known database signatures in this list of processes and services, to create database CIs.

Mapping ports to processes can require specific privileges depending on the operating system in use. If the necessary privileges are not available, this job attempts to create database CIs using the available information. However, details may be missing, for example, the database port. In such cases, you may need to run the job again after entering new credentials with the necessary privileges. For details on adding credentials, see "Credentials Data Methods" in the *HP Universal CMDB Developer Reference Guide*.

After identifying databases using the above information, this job attempts to retrieve additional information on configured (but not running) instances from registry keys (on Microsoft Windows only) and by parsing well known configuration files.

Trigger Query



NOT Reference to the credentials dictionary entry Is null

Adapter

This job uses the **Database Connections by Shell** adapter

- Input query: None
- CI Attributes conditions:

CI	Attribute Value
Shell	NOT Reference to the credentials dictionary entry is null.
IpAddress	NOT IP Probe Name Is null

- Adapter Parameters

Parameter	Description
discover_mssql. true	DFM discovers Microsoft SQL database servers.
discover_oracle. true	DFM discovers Oracle database servers.
filterByDiscoveredProcesses	This parameter should always be set to **false** because this script uses out-of-the-box process discovery on some platforms, and database processes are not included in the filters. However, since this job does not create Process CIs, setting this parameter to **false** has no adverse effects.
use_lsof	Since process to port mapping on Solaris and AIX platforms requires root privileges, set this flag to **true** if the LSOF program is available on these platforms. Using LSOF does not require root privileges.
use_sudo	Since process to port mapping on some UNIX platforms requires elevated privileges, set this flag to **true** if **sudo** is configured for **netstat**, **ps**, **pfiles**, **kdb**, or **lsof**.

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node
- Oracle

- SQL Server
- Unix
- Windows

Note: To view the topology, see "Topology" on page 405.

DB Connection by WMI Job

This section includes:

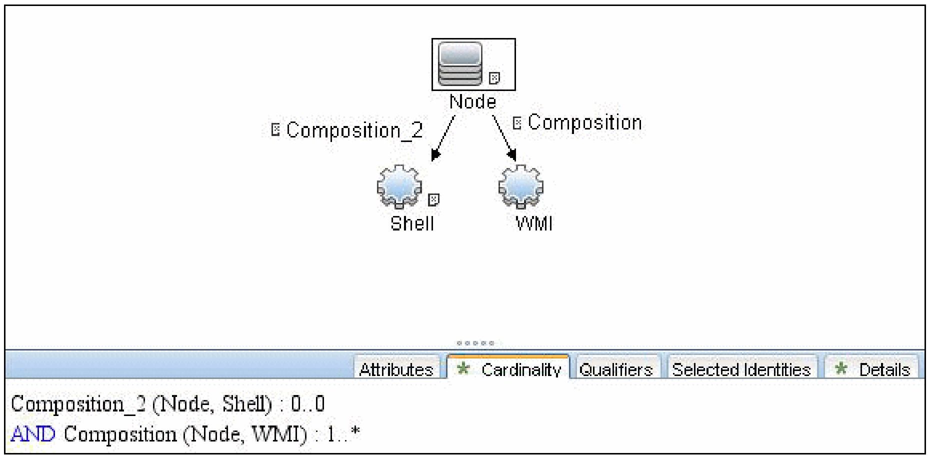
- "DB Connection by WMI Job" above
- "DB Connection by WMI Job" above
- "DB Connection by WMI Job" above
- "DB Connection by WMI Job" above

Discovery Mechanism

Similarly to the **DB Connections by Shell** job, this job attempts to create a list of processes and services, and parses them for database signatures.

Since an agent does not have access to output of commands such as **netstat**, this job is limited in that the listener ports of database servers are not always identified. Port information for databases such as Microsoft SQL Server is available in the Windows registry, and this job queries that information when connected through WMI.

Trigger Query



Composition_2 (Node, Shell) : 0..0
AND Composition (Node, WMI) : 1..*

Adapter

This job uses the **Database Connections by Agent** adapter.

- Input query: None
- Adapter parameters:

Parameter	Description
discover_mssql. true	DFM discovers Microsoft SQL database servers.
discover_oracle. true	DFM discovers Oracle database servers.

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node
- Oracle
- SQL Server
- Unix
- Windows

Note: To view the topology, see "Topology" on page 405.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Database Connections by Host Credentials discovery.

- **DB Connections by WMI discovery:** To improve performance, the trigger query for the DB Connections by WMI job has been disabled by default and you should manually select servers against which this job should run.

Chapter 30: IBM DB2 Database Discovery

This chapter includes:

Supported Versions

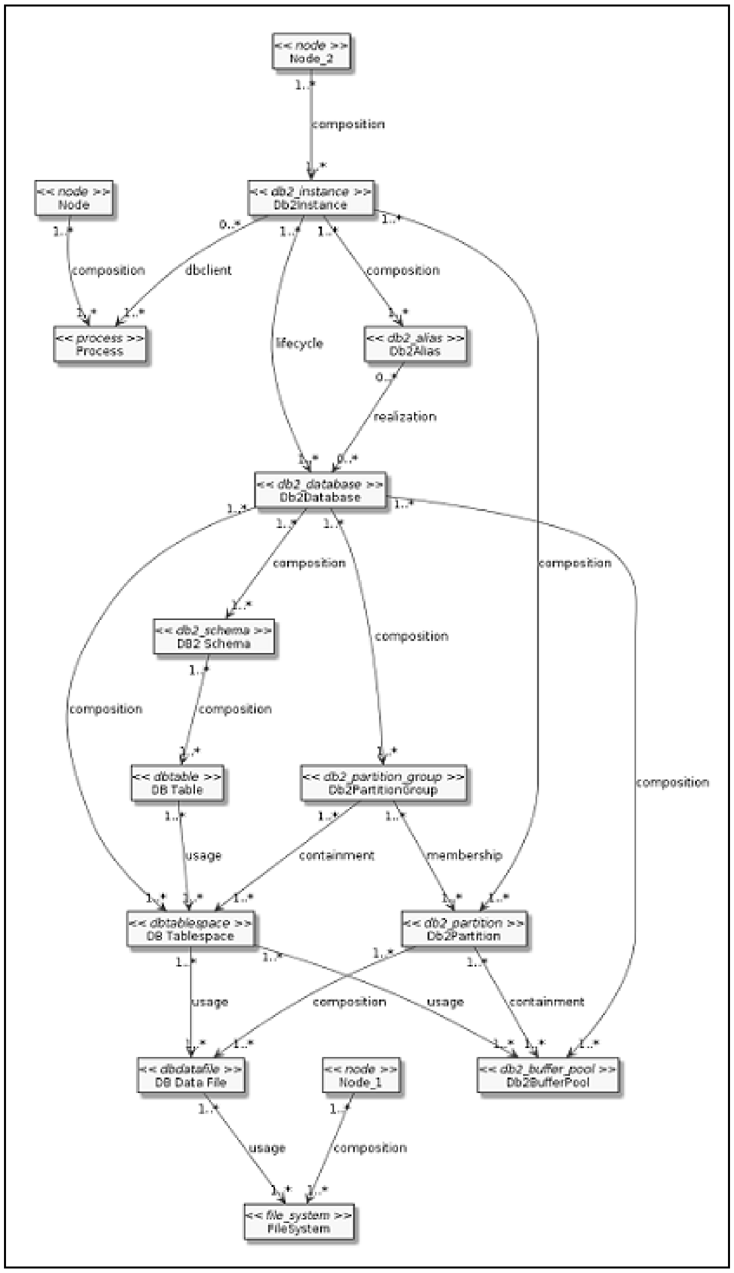
This discovery supports the following versions:

IBM DB2 Universal Database (UDB) versions 9.1, 9.5, 9.7, 9.8, and 10.1; for Linux, Unix and Windows platforms.

IBM DB2 Topology

The diagram below depicts the topology of the IBM DB2 Server view. It shows a host (Node_2) on which IBM DB2 Server and its resources are installed, the processes that communicate with the server (connected by DB Client links), and tablespace containers modeled as a DB Data File.

Note: For a list of discovered CITs, see "DB2 Universal Database Connection by SQL Job" on page 426, "DB2 Topology by SQL Job" on page 423, and "Databases TCP Ports Job" on page 419.



How to Discover Full DB2 Topology

This task discovers IBM DB2 Server databases and their components on the network, and includes the following steps.

1. **Prerequisite - Set up protocol credentials**

   - **For SQL-based Discovery:** The IBM DB2 SQL-based Discovery uses the Generic DB Protocol (SQL).

     When setting up protocol credentials:

     - In the Database Type box, choose **db2**.
     - Verify the user name, password, and port used by IBM DB2 Server.

       For more information on configuring the Generic DB Protocol (SQL), see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   - **For Shell-based Discovery:**

     - Configure the appropriate shell protocol. For more information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.
     - All possible ports used by IBM DB2 Databases must be added to **portNumberToPortName.xml** file. Make sure that **portName="db2"**.

2. **Prerequisites - Miscellaneous**

   - To perform an IBM DB2 discovery, copy the following files from the directory **<db2_home_directory>\IBM\SQLLIB\java** (for windows) or **/opt/ibm/db2/<version>/java** (for UNIX-like systems) to the Data Flow Probe machine:

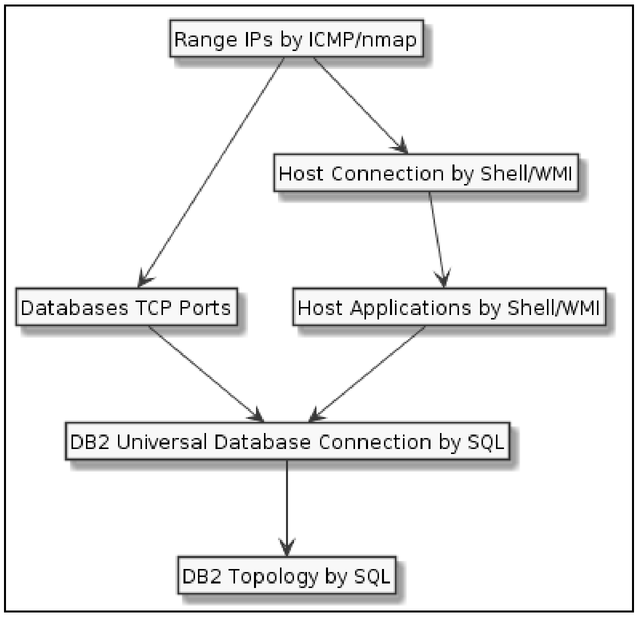
     - **db2java.zip**
     - **db2jcc.jar**
     - **db2jcc_license_cisuz.jar**
     - **db2jcc_license.jar**

     On the Data Flow Probe machine, place the files in the following folder: **<hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\db\db2** and restart the Data Flow Probe.

3. **Run the discovery**

   The jobs that you need to execute to perform this discovery are determined by whether shell access is provided to DB2 destinations. If shell access is provided, then perform discovery according to the instructions in the section below "DB2 Discovery with Shell Access Provided to DB2 Destinations" below. If shell access is not provided to DB2 destinations, perform discovery according to the instructions in the section below "DB2 Discovery without Shell Access Provided to DB2 Destinations" on the next page.

The two possible approaches to discovering full DB2 Topology are depicted in the following diagram:


Range IPs by ICMP/nmap
Host Connection by Shell/WMI
Databases TCP Ports
Host Applications by Shell/WMI
DB2 Universal Database Connection by SQL
DB2 Topology by SQL

DB2 Discovery with Shell Access Provided to DB2 Destinations

Note: When shell access is provided to DB2 destinations, DB2 Discovery can be performed in either shallow mode or deep mode. To perform a shallow discovery where the basic topology is discovered (Db2Instance, Db2Database, and Db2Alias CITs), carry out steps a-c below. To perform a deep discovery, carry out all the steps below.

To perform the DB2 Discovery:

a. In the Universal Discovery window, execute the **Range IPs by ICMP** job. This job discovers all reachable IPs.

b. Execute the job **Host Connection By Shell.** Discovers host's connectivity by shell protocol to the DB2 servers.

c. Execute the job **Host Applications by Shell.** Discovers DB2 instances, their databases and aliases.

d. To perform a deeper discovery, execute the following jobs in the order listed:

   i. **DB2 Universal Database Connection by SQL.** This job discovers the **credentials_id** attribute of **Db2Databases** and **Db2Aliases**. The discovered **credentials_ids** are used in the **DB2 Topology By SQL** job (for the list of discovered CITs see "Discovered CITs" on page 435).

   Note: This job is triggered when one of the following conditions is true:

   - A **Node** has **Db2Instance** and **Db2Aliases** that are not linked with a realization link to any **Db2Databases** (discovered by the **Host Applications by Shell** job.
   - A **Db2Database**has DB2 **IpServiceEndpoint** (discovered by the **Host Applications by Shell** job).
   - A **Node** is linked to DB2 **IpServiceEndpoint** (discovered by the **Host Applications by Shell** job).

     DB2 **IpServiceEndpoint** is an endpoint with **db2** in the **service_names** attribute or with the **ip_service_name** attribute equal to **db2**.

   ii. **DB2 Topology By SQL**. This job discovers the full DB2 topology, including the CITs listed in "Discovered CITs" on page 432.

   Note: This job is triggered when one of the following conditions is true:

   - A **Db2Alias** with the **credentials_id** attribute is reported.
   - A **Db2Database** with the **credentials_id** attribute is reported.

DB2 Discovery without Shell Access Provided to DB2 Destinations

To perform the DB2 Discovery:

a. In the Universal Discovery window, execute the **Range IPs by ICMP** job. This job discovers all reachable IPs.

b. Execute the job **Databases TCP Ports**. This job discovers the DB2 **IpServiceEndpoint**, which is a trigger for the **DB2 Universal Database Connection by SQL** job.

c. Execute the following jobs in the order listed:

   - **DB2 Universal Database Connection by SQL.** This job discovers **credentials_ids** for both **Db2Databases** and **Db2Aliases**. The discovered **credentials_ids** are used in the **DB2 Topology By SQL** job (for the list of discovered CITs see "Discovered CITs" on page 435).

     Note: This job is triggered when a **Node** is linked to DB2 **IpServiceEndpoint** (discovered by the **Databases TCP Ports** job). DB2 **IpServiceEndpoint** is an endpoint with **db2** in the **service_names** attribute or with the **ip_service_name** attribute equal to **db2**.

   - **DB2 Topology By SQL**. This job discovers the full DB2 topology, including the CITs listed in "Discovered CITs" on page 432.

     Note: This job is triggered when one of the following conditions is true:

     - A **Db2Alias** with the **credentials_id** attribute is reported.
     - A **Db2Database** with the **credentials_id** attribute is reported.

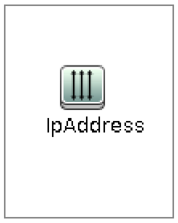
For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Databases TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host's ports.
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).

Name	Default Value	Description
ports	**For JEE TCP Ports job:** • weblogic, weblogicSSL, websphere_jmx, rmi **For Database TCP Ports job:** • oracle, db2, sybase, sql, mysql **For SAP TCP Ports job:** • sap, sap_jmx, sap_http, sap_https **For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.

Name	Default Value	Description
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

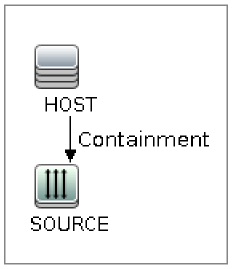
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

DB2 Topology by SQL Job

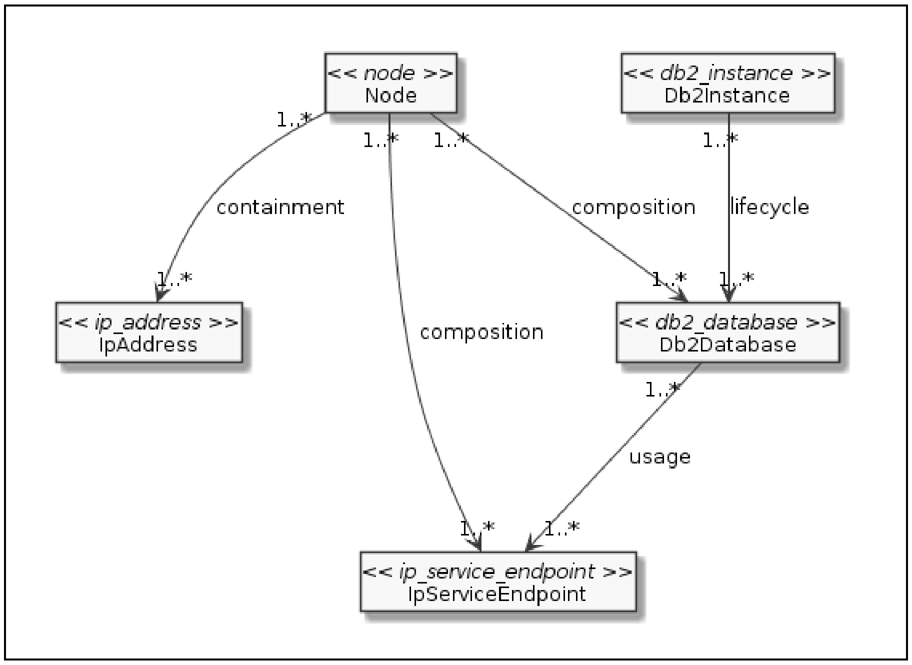
This job discovers the physical elements within a DB2 database.

Adapter

ID: **SQL_APP_Dis_Db2**

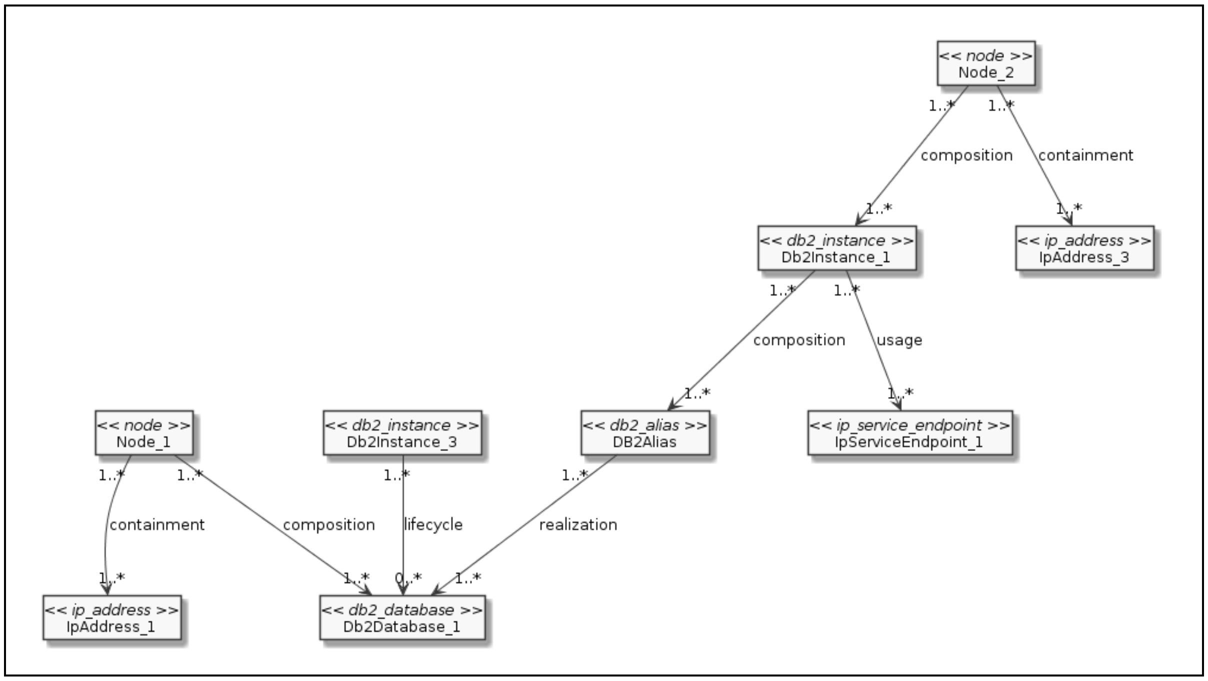
Trigger TQL

db2withuser.xml



In the diagram above, **Db2Database** should have reported credentials dictionary entry(**credentials_id**).

db2_alias_with_user.xml



In the diagram above;

- **Db2Database_1** should not have reported credentials dictionary entry(**credentials_id**).
- **Db2Alias** should have reported credentials dictionary entry(**credentials_id**).

Parameters

None

Prerequisites

- **Set up credentials.**

  This job uses SQL credentials defined for the DB2 database.

Discovery Flow

If a connection is established, discovery occurs in the following order:

1. This job retrieves DB2 major and minor versions (the `SERVICE_LEVEL` field of `TABLE (sysproc.env_get_inst_info())`).
2. This job instantiates the discoverer of the appropriate DB2 version (the 9x discoverer is used if the discoverer of the provided version is not available).
3. This job retrieves the full DB2 version information (`RELEASE_NUM, SERVICE_LEVEL, BLD_LEVEL, PTF, FIXPACK_NUM` fields of `TABLE(sysproc.env_get_inst_info())`) and the full version is reported to the `application_version` attribute of **DB2**.
4. This job retrieves **DB Data File**(`SYSIBMADM.CONTAINER_UTILIZATION`). **FileSystem** is also modeled if **DB Data File** is located on a Windows host (the disk name of the path is taken from the path of **DB Data File** and assigned to the `mountpoint` attribute of **FileSystem**).
5. This job retrieves opened sessions (`TABLE(SNAP_GET_APPL_INFO(db_name, partition_ number))`).
   - The address and port of the client are parsed from `APPL_ID` field.
   - `APPL_NAME` is used as a process name.
   - Sessions with client `APPL_NAME=db2jcc_application` or `APPL_NAME=db2jccThread` are considered to be default client name and not reported.

6. This job retrieves the following CITs:
   - **Db2 Schemas** (`SYSCAT.SCHEMATA`)
   - **Db2PartitionGroups** (`SYSCAT.DBPARTITIONGROUPS`)
   - **Db2Partitions** (`TABLE(DB_PARTITIONS())`)
   - **Db2Partition** to **Db2PartitionGroup** relation (`SYSCAT.DBPARTITIONGROUPDEF`)
   - **Db2BufferPools**(`SYSCAT.BUFFERPOOLS`). Data is also taken from the table `SYSCAT.BUFFERPOOLDBPARTITIONS` to report correctly customized **Db2BufferPools**.
   - **DB Table** (`SYSCAT.TABLES`). Only `Table (untyped)` and `Typed table` types are discovered.

Note: To view the topology, see "IBM DB2 Topology" on page 414.

DB2 Universal Database Connection by SQL Job

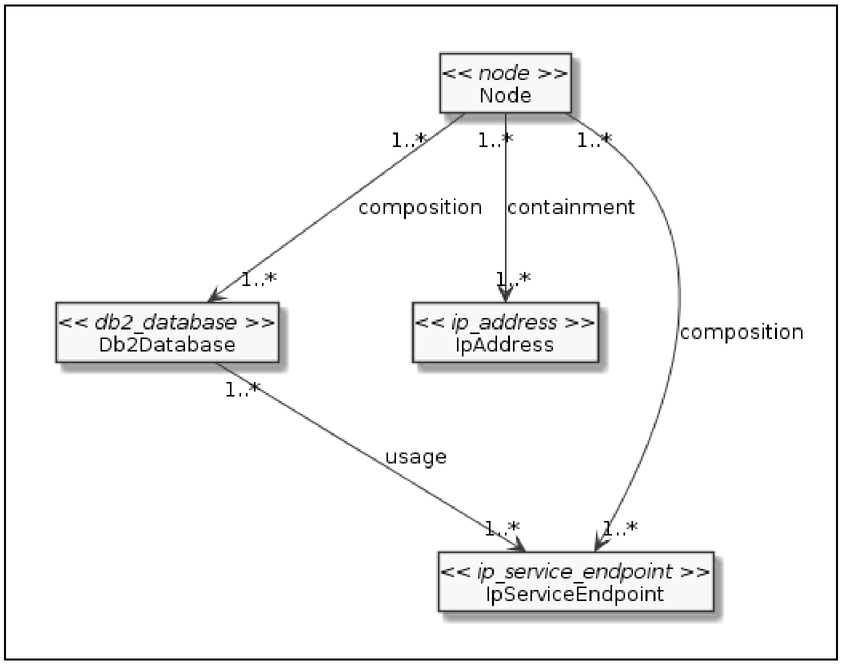
This job discovers DB2 databases using the SQL protocol.

Adapter

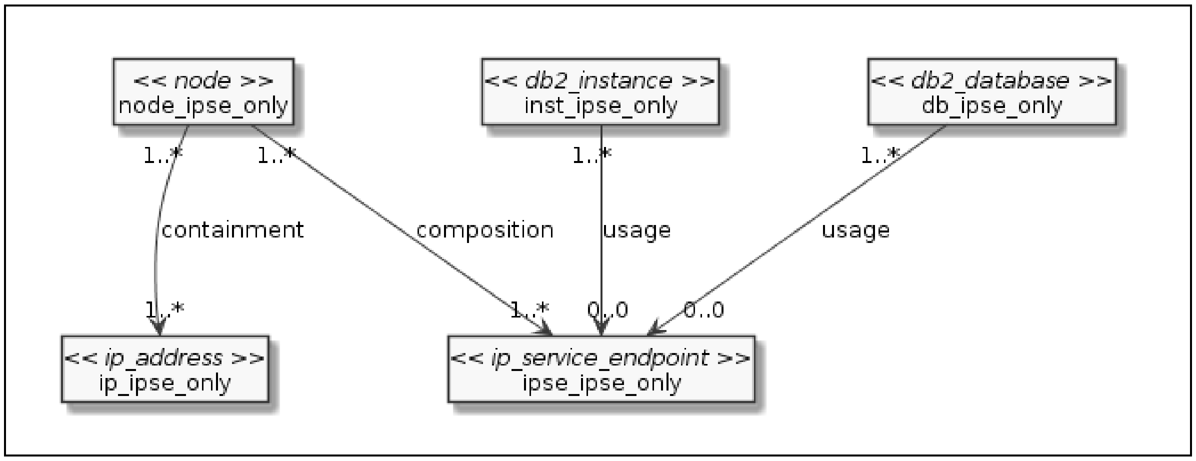
ID: **SQL_NET_Dis_Connection_DB2**

Trigger TQL

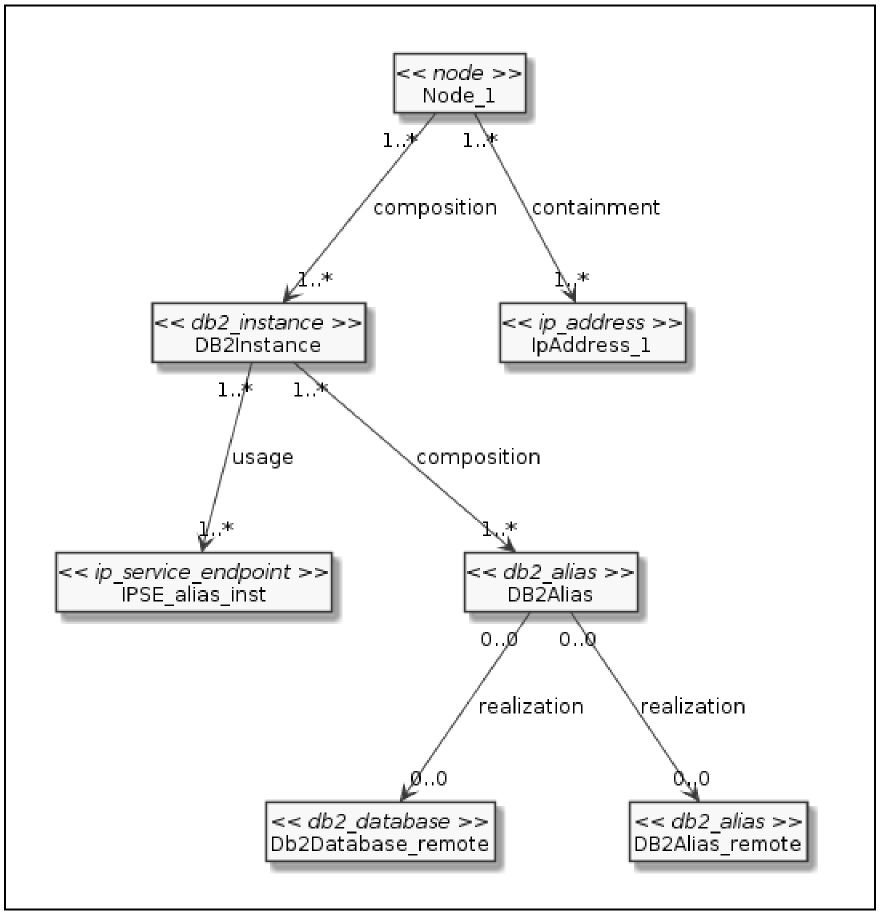
db2_db_port_sa.xml



db2_ipse_only.xml



db2_alias_no_realizaiton_.xml



Parameters

None

Prerequisites

- **Set up credentials.**

  This job uses SQL credentials defined for the DB2 database.

  Note: Port and database name are optional for DB2 credentials and if they are not set they are considered as candidates to be used during connection.

Discovery Flow

This job iterates over available Generic DB Protocol credentials of type **db2**. Credential entries are considered applicable if they do not contain port or database name information. If they do contain port or database name information, they are additionally compared with the port and database name provided from the triggered CI data (**port** and **db_name**, respectively).

As a result of this job, **Db2Database** or **Db2Alias** (depending on the trigger) is modeled with the appropriate **credentials_id** field.

DB2 Topology by SQL Adapter

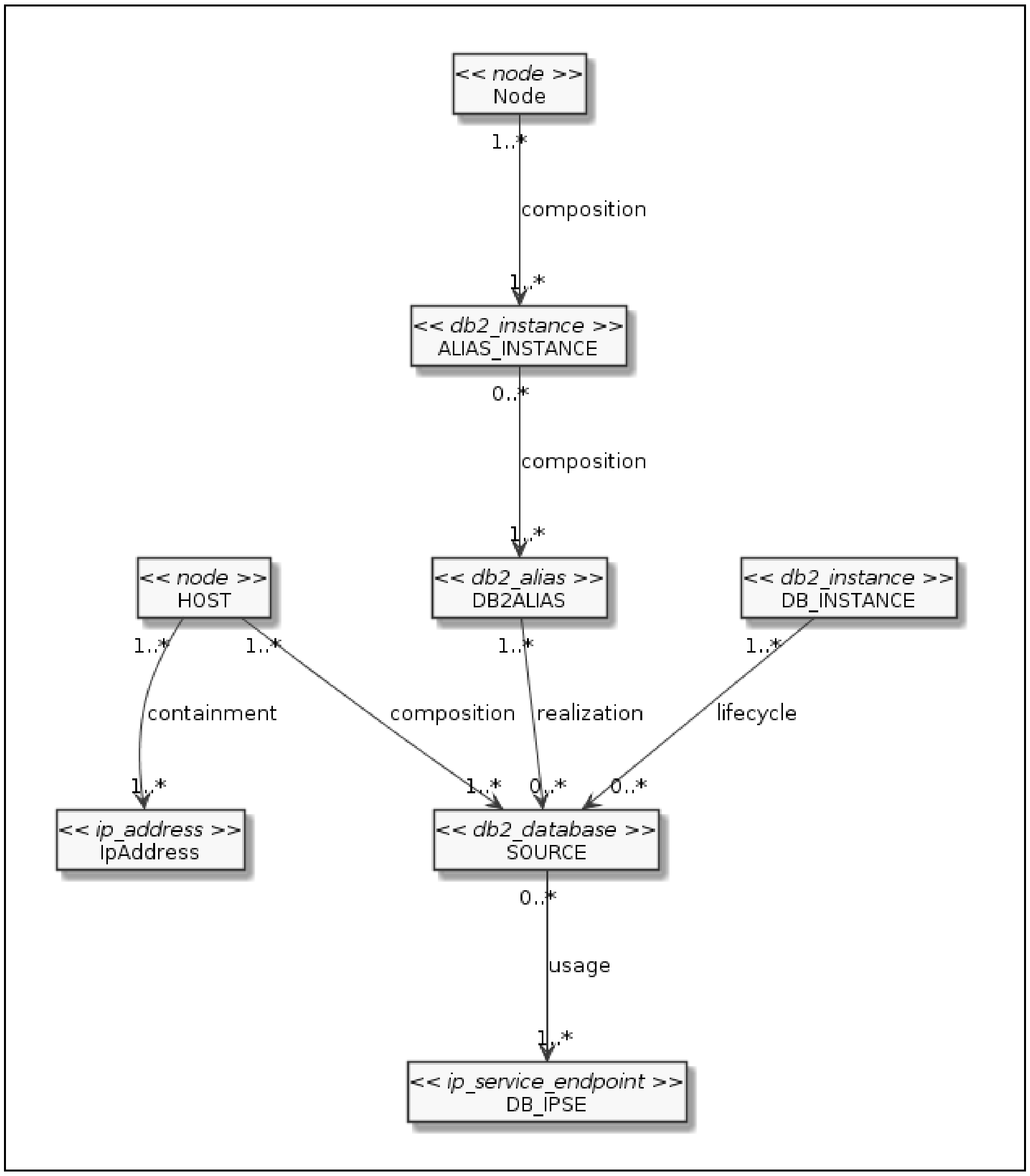
ID

`SQL_APP_Dis_Db2`

Input CIT

`Db2Database`

Input TQL



Triggered CI Data

Name	Value	Description
alias_ inst_ip_ address	${ALIAS_ INSTANCE.application_ ip:!NA!}	The IP address of the DB2 instance owning the corresponding alias, in the case when the trigger is **Db2Alias** with **credentials_id** reported.

Name	Value	Description
alias_inst_ipse_ip_address	${ALIAS_INSTANCE_IPSE.bound_to_ip_address:!NA!}	The **IpServiceEndpoint** IP address used for establishing the SQL connection, in the case when the trigger is **Db2Alias** with **credentials_id** reported
alias_inst_ipse_port	${ALIAS_INSTANCE_IPSE.network_port_number:!NA!}	IThe **pServiceEndpoint** port used for establishing the SQL connection, in the case when the trigger is **Db2Alias** with **credentials_id** reported.
alias_inst_port	${ALIAS_INSTANCE.application_port:!NA!}	The port number used by the DB2 instance owning the corresponding alias, in the case when the trigger is **Db2Alias** with **credentials_id** reported.
alias_cred_id	${DB2ALIAS.credentials_id:!NA!}	The **credentials_id** used for establishing the SQL connection, in the case when the trigger is **Db2Alias** with **credentials_id** reported.
alias_name	${DB2ALIAS.name:!NA!}	The alias name used as a DB name for establishing SQL connection, in the case when the trigger is **Db2Alias** with **credentials_id** reported.
db_instance_id	${DB_INSTANCE.root_id:!NA!}	The **Db2Instance** CMDB ID, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_ipse_id	${DB_IPSE.root_id:!NA!}	The **IpServiceEndpoint** CMDB ID, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_ipse_ip_address	${DB_IPSE.bound_to_ip_address:!NA!}	The **IpServiceEndpoint** IP address used for establishing the SQL connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_ipse_port	${DB_IPSE.network_port_number:!NA!}	The **IpServiceEndpoint** port used for establishing the SQL connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.
host_id	${HOST.root_id}	The Node CMDB ID containing the target DB2 database.
db_node_ip_address	${IpAddress.ip_address:!NA!}	The Node IP address of **Db2Database** used for establishing a connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.

Name	Value	Description
db_cred_ id	${SOURCE.credentials_ id:!NA!}	The DB2Database **credentials_id** used for establishing the SQL connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_id	${SOURCE.root_id}	The **Db2Database** CMDB ID used to restore a CI, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_name	${SOURCE.name}	The **Db2Database** name used for establishing the SQL connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.
db_port	${SOURCE.application_ port:!NA!}	The **Db2Database** port used for establishing the connection, in the case when the trigger is **Db2Database** with **credentials_id** reported.

Used Scripts

- db2_topology_by_sql.py

Discovered CITs

- Composition
- Containment
- DB Data File
- DB Table
- DB Tablespace
- DB2 Schema
- Db2Alias
- Db2BufferPool
- Db2Database
- DB2Instance

- Db2Partition
- Db2PartitionGroup
- IpAddress
- IpServiceEndpoint
- Membership
- Node
- Process
- Resource
- Usage

Parameters

Name	Type	Description
discoverTables	boolean	This flag indicates whether the job should discover tables.
discoverSystemTables	boolean	This flag indicates whether job should discover system tables. If **discoverTables** is false, this flag is considered false as well.

DB2 Connection by SQL Adapter

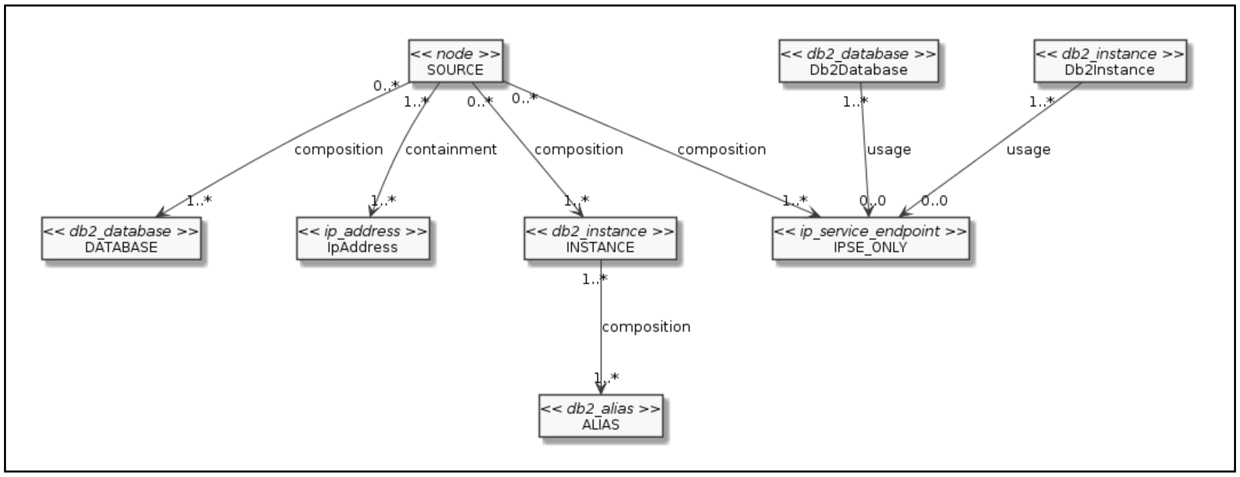
ID

`SQL_NET_Dis_Connection_DB2`

Input CIT

Node

Input TQL



<< node >> SOURCE
<< db2_database >> Db2Database
<< db2_instance >> Db2Instance
<< db2_database >> DATABASE
<< ip_address >> IpAddress
<< db2_instance >> INSTANCE
<< ip_service_endpoint >> IPSE_ONLY
<< db2_alias >> ALIAS
composition
containment
usage
0..*
1..*
0..0

Triggered CI Data

Name	Value	Description
alias_container	${ALIAS.root_container:!NA!}	A list of **Db2Alias** container CMDB ids, in the case when the trigger is **Db2Alias** without an outgoing realization link.
alias_id	${ALIAS.root_id:!NA!}	A list of **Db2ALias** CMDB ids, in the case when the trigger is **Db2Alias** without an outgoing realization link.
alias_name	${ALIAS.name:!NA!}	A list of **Db2ALias** names, used for establishing connections, in the case when the trigger is **Db2Alias** without an outgoing realization link.
db_application_ip	${DATABASE.application_ip:!NA!}	A list of **Db2Database** IP addresses, used for establishing connections, in the case when the trigger is **Db2Database** with db2 **IpServiceEndpoint**.
db_application_port	${DATABASE.application_port:!NA!}	A list of **Db2Database** ports, used for establishing connections, in the case when the trigger is **Db2Database** with db2 **IpServiceEndpoint**.
db_id	${DATABASE.root_id:!NA!}	A list of **Db2Database** CMDB ids from which CIs are restored, in the case when the trigger is **Db2Database** with db2 **IpServiceEndpoint**.

Name	Value	Description
db_name	${DATABASE.name:!NA!}	A list of **Db2Database** names, used for establishing connections, in the case when the trigger is **Db2Database** with db2 **IpServiceEndpoint**.
credentials_id	${DATABASE.credentials_id:!NA!}	A list of Db2Database **credentials_ids**, used for establishing connections, in the case when the trigger is **Db2Database** with db2 **IpServiceEndpoint**.
inst_application_port	${INSTANCE.application_port:!NA!}	A list of **Db2Instance** ports, used for establishing connections, in the case when the trigger is **Db2Alias** without an outgoing realization link.
inst_id	${INSTANCE.root_id:!NA!}	A list of **Db2Instance** CMDB ids, in the case when the trigger is **Db2Alias** without an outgoing realization link.
ipse_only_id	${IPSE_ONLY.root_id:!NA!}	A list of **IpServiceEndpoint** CMDB IDs from which CIs are restored, in the case when the trigger is db2 **IpServiceEndpoint**.
ipse_only_port	${IPSE_ONLY.network_port_number:!NA!}	A list of **IpServiceEndpoint** ports, used for establishing connections, in the case when trigger is db2 **IpServiceEndpoint**.
ip_address	${IpAddress.ip_address}	The current **Node** IP address.
host_id	${SOURCE.root_id}	The current Node CMDB id, from which CIs are restored.

Used Scripts

- db2_connection_by_sql.py

Discovered CITs

- Composition
- Db2Alias
- Db2Database
- Db2Instance
- IpServiceEndpoint

- Node
- Usage

Parameters

None

Application Signatures and Plugins

Application Signatures

The Db2 package has two signatures:

- **IBM DB2 on Unix.** Discovers the Db2Instance by presence of the **db2sysc** process. If the port is recognized according to configuration in **PortNumberToPortName**, it is reported with the service name db2. By default the configuration file for port to name mapping declares 50000 and 6789 ports.

- **IBM DB2 on Windows.** Discovers the Db2Instance by presence of **db2syscs.exe** process. If the port is recognized according to the configuration in PortNumberToPortName, it is reported with service name db2. By default the configuration file for port to name mapping declares 50000 port.

Plugins

Additional topology is reported by the following plugins:

- **db2_instances_on_windows.** Discovers **Db2Databases**, **Db2Instances**, and **Db2Aliases** on a Windows host.

- **db2_instances_on_unix.** Discovers **Db2Databases**, **Db2Instances**, and **Db2Aliases** on a Unix host.

`PortNumberToPortName` Configuration

The **portNumberToPortName.xml** file contains all the possible ports that are used by IBM DB2 Databases across the target discovery range. These ports are named **"db2"**.

Troubleshooting and Limitations

Troubleshooting

Problem: If the target DB2 instance port was not added to the **portNumberToPortName.xml** file (which means that it is not recognized as **db2**), the "Multiple Match" warning may appear in UCMDB UI (which means that the target CI is not reported) after running of 'Host Applications by Shell' job.

Solution: Add the target port to **portNumbertToPortName.xml** file as a db2 port entry.

Limitations

Limitation when performing DB2 Discovery without Shell Access

The DB2 platform allows specifying a network service name as a listening port for the instance. This network service name is an alias that should be resolved with an appropriate mapping file (/etc/services on Unix and %SystemRoot%\system32\drivers\etc\services on Windows) and needs shell access to get the content of this file. In the case when there is no shell access and only SQL-based access, it is not possible to expand the service name string to a port number. This causes **IpServiceEndpoint** to not be reported for the connected database. The only workaround for this is to use a real port number when configuring the DB2 instance instead of the service name.

Chapter 31: HP NonStop Discovery

This chapter includes:

Overview

Since its inception in the mid-1970s, the HP NonStop server has held an important role in helping global business run smoothly, effectively, and successfully. Today, NonStop servers process the overwhelming majority of credit card, automated teller machine (ATM), and securities transactions. The world's leading enterprises rely on NonStop servers, including 106 of the 120 largest stock and commodity exchanges and 135 public telephone companies. Innovative solutions based on the NonStop platform help customers achieve a competitive advantage in multiple industry sectors, including financial services, telecommunications, healthcare, retail, public sector, and manufacturing. Based on studies by The Standish Group, the NonStop server delivers the lowest total cost of ownership (TCO) in the industry for servers of its class.

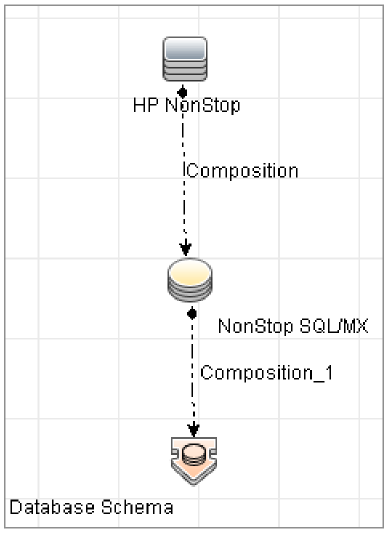
Supported Versions

This discovery solution supports:

- HP NonStop H06.x
- NonStop SQL/MX 2.3
- NonStop SQL/MP H01 series.

Note: The discovery is expected to work on all available versions of HP NonStop.

Topology


HP NonStop
Composition
NonStop SQL/MX
Composition_1
Database Schema

How to Discover HP NonStop

The following steps describes how to perform HP NonStop discovery.

1. **Prerequisites**

   Before starting the discovery, ensure that the discovery user was granted all of the required permissions to run the following commands:

   - **gtacl -p scf info lif '$zzlan.*'**
   - **gtacl -p scf info subnet '$*.*'**
   - **mxci**
     - **set schema nonstop_sqlmx_<node_name>.system_schema**
     - **select cat_name, cat_uid from catsys**
     - **select schema_name, cat_uid from schemata**
   - **gtacl -p sqlci**
     - **fileinfo $system.system.sqlci2, detail**
     - **select catalogname from <catalog_file_name>.catalogs**

2. **Set up network and protocol credentials**

   The HP NonStop discovery solution is based on the SSH protocol. The corresponding credentials must be provided in order to use this protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

3. **Run the Discovery**

   To discover the topology:

   a. Run the **Range IPs by ICMP** or **Range IPs by nmap** job to discover the HP NonStop system IP addresses.

   b. Run the **Host Connection by Shell** job to discover the HP NonStop system with the SSH agent and networking topology connected.

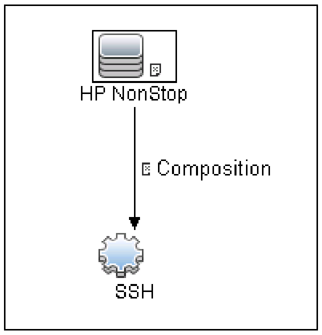
c. Run the **HP NonStop Topology by Shell** job to discover the shallow SQL MP/MX topology.

HP NonStop Topology by Shell Job

This section includes:

- "HP NonStop Topology by Shell Job" above
- "HP NonStop Topology by Shell Job" above
- "HP NonStop Topology by Shell Job" above

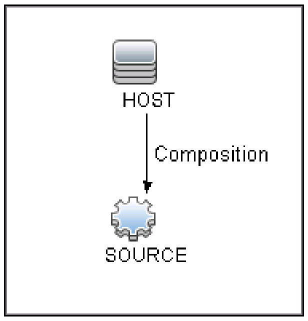
Trigger Query



Adapter

This job uses the **hp_nonstop_topology_by_shell** adapter.

- Input CIT: **SSH**
- Input Query:



- Used Scripts

- hpnonstop_topology_by_shell.py
- hpnonstop_networking.py
- TTY_Connection_Utils.py

Note: This job may also use library scripts supplied in the AutoDiscoveryContent package.

- Created/Changed Entities:

Entity Name	Entity Type	Entity Description
hp_nonstop	CIT	New CIT which represents HP NonStop System
nonstop_sql_mx	CIT	New CIT which represents SQL/MX database
HP NonStop Topology by Shell	Job	New topology job
hp_nonstop_ topology_by_shell	Adapter	Discovery adapter
Host_Connection_ By_Shell	Adapter	Adding HP NonStop support caused the adapter used by Host Connection by Shell job to change.
hpnonstop_ topology_by_ shell.py	Script	Discovery Jython script
hp_nonstop_ shell.xml	TQL	Trigger TQL
TTY_Connection_ Utils	Script	Main script used by Host Connection by Shell job has changed in order to support HP NonStop systems
hp_nonstop_ networking.py	Script	Jython script that discovers HP NonStop networking information

Discovered CITs

- Composition

- Database
- Database Schema
- HP NonStop
- NonStop SQL/MX

HP NonStop Discovery Commands

This section describes each of the commands used by HP NonStop discovery.

This section includes:

Command: gtacl -p scf info lif ';$zzlan.*';

- **Sample Output**

```
SCF - T9082H01 - (16JUL10) (30MAR10) - 11/08/2010 01:32:10 System \NON_STOP_
SYSTEM
(C) 1986 Tandem (C) 2006 Hewlett Packard Development Company, L.P.
SLSA Info LIF
```

```
Name              Associated Object     MAC Address         Type
$ZZLAN.LANA       G4SA0.0.A             01:01:01:01:01:01   Ethernet
$ZZLAN.LANB       G4SA0.0.B             02:02:02:02:02:02   Ethernet
$ZZLAN.LANC       G4SA0.0.C             03:03:03:03:03:03   Ethernet
$ZZLAN.LAND       G4SA0.0.D             04:04:04:04:04:04   Ethernet
Total Errors = 0    Total Warnings = 0
```

- **Modeled CITs: Interface**

Attribute	Value	Comment
Name	**LANA**	
Interface MAC Address	**01:01:01:01:01:01**	
Interface Description	**G4SA0.0.A**	

Command: gtacl -p scf info subnet ';$*.*';

- **Sample Output (partial)**

```
SCF - T9082H01 - (16JUL10) (30MAR10) - 11/08/2010 04:05:58 System \MEASYOS
(C) 1986 Tandem (C) 2006 Hewlett Packard Development Company, L.P.
TCPIP Info SUBNET \MEASYOS.$ZSM1.*
Name     Devicename     *IPADDRESS       TYPE     *SUBNETMASK  SuName   QIO *R
#SN01    \MEASYOS.LANC   10.10.10.10   ETHERNET  %HFFFFFC00           ON  N
#LOOP0                   127.0.0.1       LOOP-BACK %HFF000000           OFF N
TCPIP Info SUBNET \MEASYOS.$ZTC0.*
Name     Devicename     *IPADDRESS       TYPE     *SUBNETMASK  SuName   QIO *R
#SN01    \MEASYOS.LANC   10.10.10.10   ETHERNET  %HFFFFFC00           ON  N
#LOOP0                   127.0.0.1       LOOP-BACK %HFF000000           OFF N
```

- **Modeled CITs: IP, Network**

Attribute	Value	Comment
IP Address	**10.10.10.10**	Only "ETHERNET" type is considered
IP Network Mask	**%HFFFFFC00**	A network mask represented in HEX format
Container	**LANC**	The name of the interface where this IP is connected to

Note: The Network CIT is also created from this command.

Command: mxci

- **Sample Output**

```
Hewlett-Packard NonStop(TM) SQL/MX Conversational Interface 2.3.4
(c) Copyright 2003, 2004-2010 Hewlett-Packard Development Company, LP.
```

- **Values Taken**

  SQL/MX version value is taken from the output. In this case, it is 2.3.4.

Command: set schema nonstop_sqlmx_measyos.system_schema;

- **Sample Output**

```
--- SQL operation complete.
```

- **Modeled CITs**

  None

Command: select cat_name, cat_uid from catsys;

- **Sample Output**

```
CAT_NAME                                                          CAT_UID
-----------------------------------------------------------------  -------------
-------
C
0101010101010101010
NONSTOP_SQLMX_MEASYOS
0202020202020202020
--- 2 row(s) selected.
```

- **Modeled CITs - NonStop SQL/MX**

Attribute	Value	Comment
Name	NonStop SQL/MX	This value is a constant
Catalog UUID	0101010101010101010	
The Database instance name	NONSTOP_SQLMX_MEASYOS	

Command: select schema_name, cat_uid from schemata;

- **Output**

```
SCHEMA_NAME                                                       CAT_UID
-----------------------------------------------------------------  -------------
-------
DEFINITION_SCHEMA_VERSION_1200
0101010101010101010
S
0202020202020202020
DEFINITION_SCHEMA_VERSION_1200
0202020202020202020
--- 7 row(s) selected.
```

- **Modeled CITs: Database Schema**

Attribute	Value	Comment
Name	DEFINITION_SCHEMA_VERSION_1200	This is the schema ID
Container	0101010101010101010	

Command: exit

- **Sample Output**

```
End of MXCI Session
```

Command: gtacl -p sqlci

- **Sample Output**

```
SQL Conversational Interface - T9191H01^ACM - (01OCT09)
(C) 1987 COMPAQ (C) 2006 Hewlett Packard Development Company, L.P.
```

Command: fileinfo $system.system.sqlci2, detail;

- **Sample Output**

```
$SYSTEM.SYSTEM.SQLCI2                8 Nov 2010,  6:22
    ENSCRIBE ( VALID SQL PROGRAM )
    CATALOG $QA1.SQL
    PROGRAM CATALOG VERSION 1
    PROGRAM FORMAT VERSION  350
    TYPE U
    FORMAT 1
    CODE 100
    EXT ( 56 PAGES, 56 PAGES, MAXEXTENTS 978 )
    ODDUNSTR
    NO AUDITCOMPRESS
    OWNER -1
    SECURITY (RWEP): NUNU
    MODIF: 21 Dec 2008, 23:22, OPEN
    CREATION DATE: 21 Dec 2008, 23:21
    LAST OPEN:  8 Nov 2010,  6:22
    EOF 364544 (0.3% USED)
    EXTENTS ALLOCATED: 4
```

- **Values Taken**

```
QA1.SQL
```

Command: select catalogname from $QA1.SQL.catalogs;

- **Sample Output**

```
CATALOGNAME
-------------------------
\MEASYOS.$QA1.H03SQLMP
\MEASYOS.$QA1.SQL
\MEASYOS.$QA2.PERSNL
\MEASYOS.$SFF04.SALES
\MEASYOS.$SGT01.INVENT
\MEASYOS.$SGT01.PERSNL
\MEASYOS.$SGT02.SALES
\MEASYOS.$SGT03.INVENT
\MEASYOS.$SYSTEM.SRK
\MEASYOS.$SYSTEM.VIMAL
--- 10 row(s) selected.
```

- **Modeled CITs: Database**

Attribute	**Value**	**Comment**
Name	NonStop SQL/MX	This value is a constant

Attribute	Value	Comment
Database instance name	$QA1.H03SQLMP	

Chapter 32: SAP HANA Database Discovery

This chapter includes:

Overview

SAP HANA (High Performance Analytic Appliance) is SAP's database technology. It is distributed as an appliance, a combination of hardware approved by SAP, and as in-memory database software.

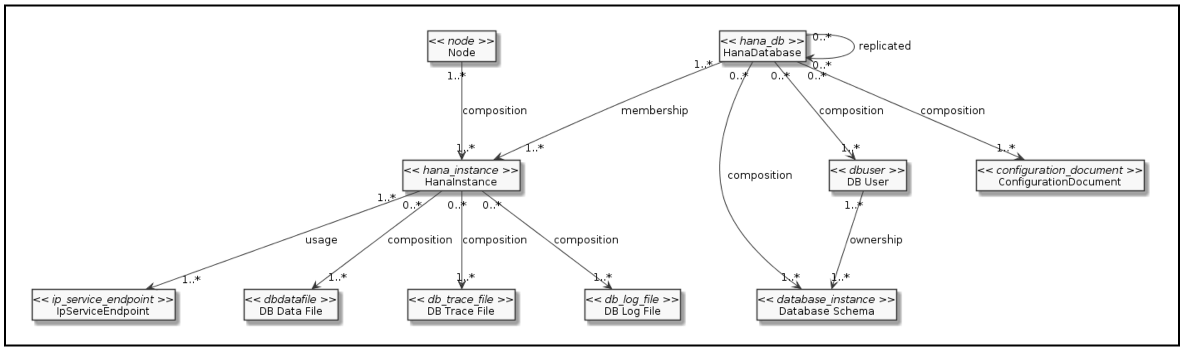
Supported Versions

This discovery supports SAP HANA 1.0 running in a UNIX environment.

Topology

The following image displays the topology of the SAP HANA Database discovery:

For a list of discovered CITs, see "Discovered CITs" on page 458.



Discovery Mechanism

Signatures

The **Hanadb** package has one signature: **SAP HanaDB**, which discovers Hana instances by the presence of a process that starts with **HDB.sap** (HDB daemon). The database name and instance number are parsed from the HDB daemon name using the **hanadb_instance_name** parsing rule.

Plugins

Additional topology is reported by the **hanadb** plugin, which discovers HanaDatabase and other Hana instances and used SQL ports that are modeled as **IpServiceEndpoints**.

How to Discover SAP HANA Database

This section describes how to discover the topology of SAP Hana Database. It includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Configure the shell protocol credential.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the hdbsql command line tool**

   Shell-based HANA discovery uses the `hdbsql` command line tool to execute SQL queries. The following paths are searched:

   - ***<installation_path>*/hdbclient/hdbsql**
   - **/usr/sap/hdbclient/hdbsql**
   - **/usr/sap/<*db_sid*>/hdbclient/hdbsql**
   - **/usr/sap/<*db_sid*>/exe/linuxx86_64/hdb/hdbsql**
   - **/usr/sap/<*db_sid*>/SYS/global/hdbclient/hdbsql**
   - **/sapmnt/<*db_sid*>/hdbclient/hdbsql**
   - **/sapmnt/<*db_sid*>/global/hdbclient/hdbsql**

   where <***db_sid***> is the SID of the HANA database.

   If the tool is not available in any of those paths, then its presence is checked in the PATH variable. Otherwise, the discovery process throws a **NoHdbsqlException** and stops discovery for the current HANA database.

3. **Configure the HDB User Store**

   To perform shallow or deep discovery, you must properly configure the HDB user store for the destination being discovered. The current discovery mechanism uses cmdb <*db_sid*> HDB User Storename, where <*db_sid*> is the SID of the HANA database.

> **Note:** The hdbsql tool requires a valid HDB user store entry to be specified with the -U option.

4. **Run the discovery**

   Activate the following jobs:

   - Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up

     To perform a shallow discovery where the basic topology is discovered (HanaDatabase, HanaInstance, and IpServiceEndpoint CITs), execute only the following two jobs and do not proceed to the following step:

     - Run the **Host Connection by Shell** job to discover a host's connectivity by shell protocol to HANA servers.
     - Run the **Host Applications by Shell** job to discover a HANA database, its instances, and the instance's IP service endpoints.

   - To perform a deep discovery, also run the **HanaDb by Shell** job to discover the topology of the target HANA Database. This job also discovers all HANA Database resources (schemas, database users, and configuration files) and HANA instances resources (data, trace files, and log files).

HanaDb by Shell Job

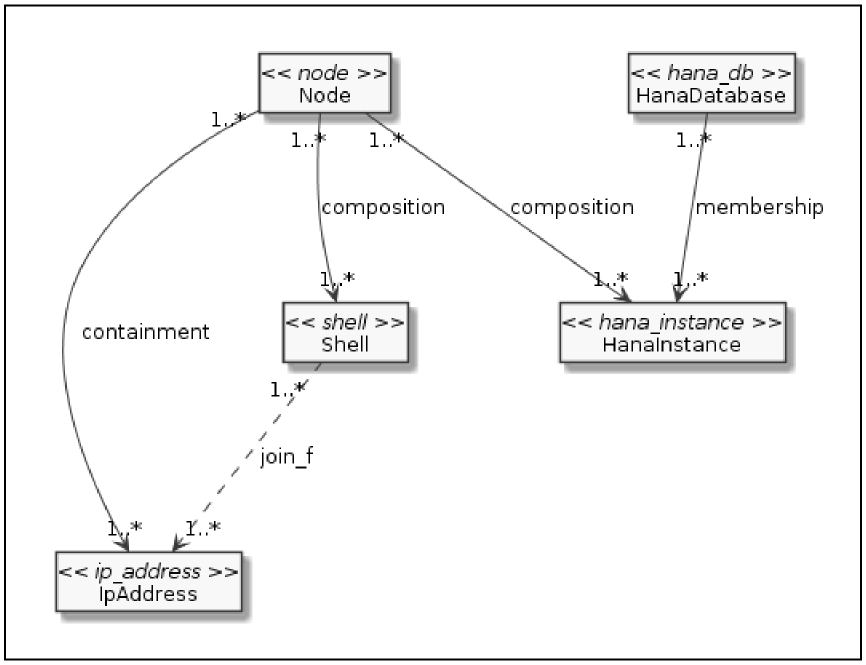
This section includes:

Adapter

This job uses the **HanaDb_by_Shell** adapter.

Trigger Query

Name: hanadb



Node Name	Condition
HanaDatabase	None
IpAddress	NOT IP Probe Name Is null
Process	Name Like ignore case %hdb.sap%
Shell	None
Node	None
HanaInstance	NOT Application Installed Path is null

Parameters

Parameters are not overridden by default, and use values from the adapter.

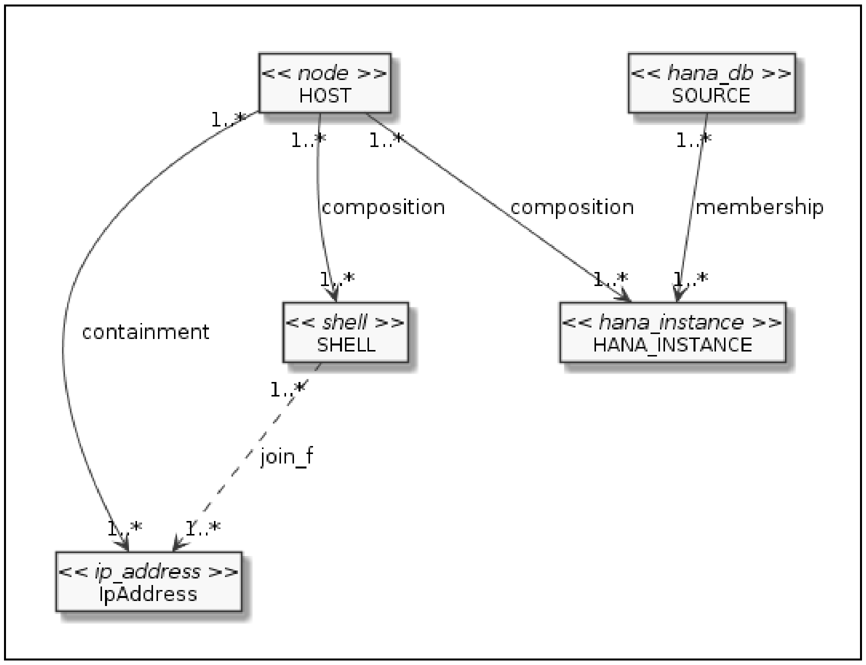
HanaDb_by_Shell Adapter

This section includes:

Input CIT

Hana Database.

Input Query



Node Name	Condition
SOURCE	None
IpAddress	NOT IP Probe Name Is null
PROCESS	Name Like ignore case %hdb.sap%
SHELL	None
HOST	NOT Application Installed Path is null

Triggered CI Data

Name	Value	Description
installpath	${HANA_ INSTANCE.application_ path}	List of installation paths for each HanaInstance connected to the current HanaDatabase.
Protocol	${SHELL.root_class}	List of protocol names, one for each existing HanaInstance shell.
credentialsId	${SHELL.credentials_id}	List of references to the credentials dictionary to be used for the connection.
ip_address	${SHELL.application_ip}	List of IP addresses for each shell connected to the HanaInstance.
sid	${SOURCE.name}	The SID of the HanaDatabase CI.
hanadb_ cmdbid	${SOURCE.root_id}	The CMDB ID of the HanaDatabase CI.

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database Schema
- DB Data File
- DB User
- DbLogFile
- DbTraceFile
- Dependency
- HanaDatabase

- HanaInstance
- IpAddress
- IpServiceEndpoint
- Node
- Ownership
- Usage

Discovery Flow

1. **Prerequisite - Set up protocol credentials**

   This discovery uses Shell protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Discovery flow**

   This adapter iterates over available Hana database instances for which the installation path attribute is not empty and
   for which at least one Shell CI is present for the host of each instance.

   If the connection is established successfully for an instance, then the **HanaDb by Shell** job attempts to discover all the resources of this instance. To simplify the discovery flow, database resources (schemas, database users, and configuration files) are also discovered during each iteration.

Chapter 33: MS-SQL Discovery

This chapter includes:

Overview

MS SQL Discovery discovers MS SQL database servers and database topology.

MS SQL database servers can be discovered either by Generic DB Protocol (SQL) or by OS credentials. MS SQL database topology can be discovered by Generic DB Protocol (SQL) only.

Supported Versions

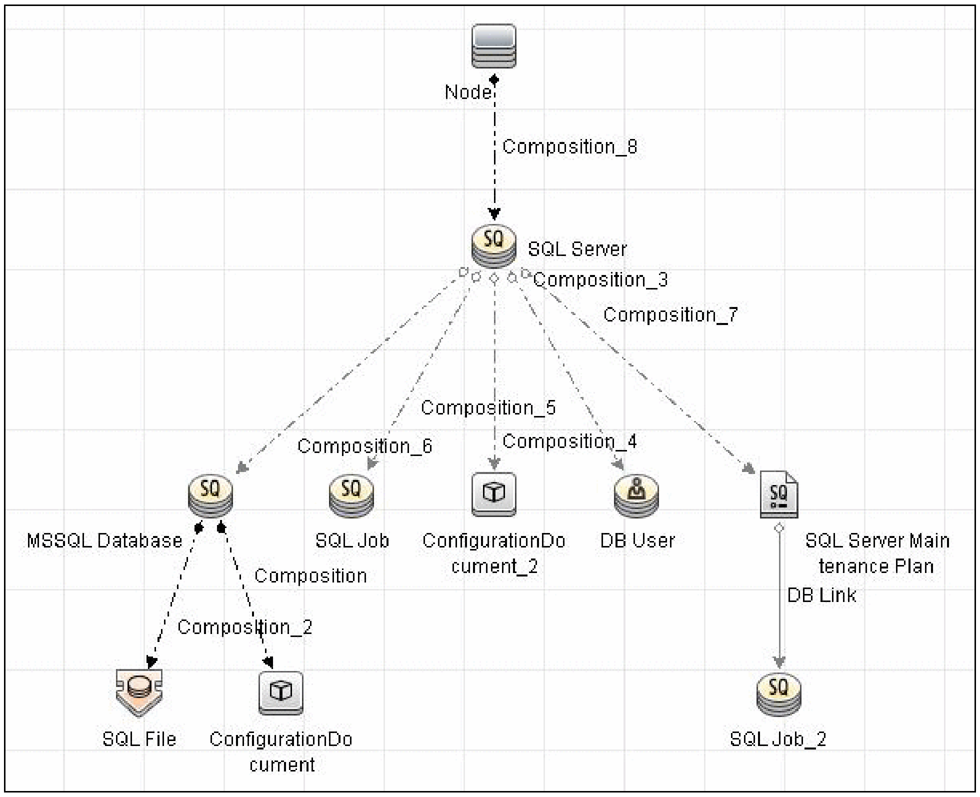
This discovery supports Microsoft SQL Server versions 2000, 2005, 2008, 2008 R2, 2012, and 2014.

Topology

The following image displays the topology of the Microsoft SQL Server Database discovery.

This view shows the hosts on which Microsoft SQL Server is installed. Microsoft SQL Server contains the databases, users, SQL jobs, and configuration files of this database, and maintenance plans.

Note: For a list of discovered CITs, see "Discovered CITs" on page 465.



How to Discover Microsoft SQL Server Database Application

This task describes how to discover the Microsoft SQL Server database application.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Microsoft SQL Server uses the Generic DB Protocol (SQL). This protocol for Microsoft SQL Server contains:

   - Microsoft SQL Server protocol; the database login and password used for authentication.
   - Microsoft SQL Server NTLM protocol; the OS login and password used for authentication.
   - Microsoft SQL Server NTLMv2 protocol; version 2 of the protocol with the OS login and password used for authentication.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Verify the user on the Microsoft SQL Server**

   Verify the user name, password, and port used by Microsoft SQL Server.

3. **Run the discovery**

   In the Universal Discovery window, activate the jobs in the following order:

   - Databases TCP Ports
   - MSSQL Server Connection by SQL
   - MSSQL Topology by SQL

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover MS SQL Server Components Using OS Credentials

1. **Run the discovery**

   The following jobs discover MS SQL Server components using OS credentials:

   - **Host Applications by Shell**
   - **Host Applications by WMI**
   - **DB connection by Shell**
   - **DB connection by WMI**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Microsoft SQL Server Database Application Discovery

Adapter

Adapter Parameters for the MSSQL Topology by SQL job

Parameter	Description
discoverConfigs	**True** (default)**.** Server configuration ('mssql database configuration.txt') is retrieved.
discoverDbUser	**False** (default). DB User entities for MS SQL Server are not retrieved.
discoverSqlFile	**False** (default). SQL File entities for MS SQL Server are not retrieved.
discoverSqlJob	**False** (default). SQL Job entities for MS SQL Server are not retrieved.
discoverStoredProcedures	**False** (default). Do not discover stored procedures from the MS SQL Database Server.

Parameter	**Description**
discoverInternalProcedures	**False** (default). Internal MS SQL stored procedures are filtered during discovery and are not reported to the UCMDB. **Note:** Internal MS SQL stored procedures come from MSDB and Master databases and begin with suffixes: **sp_**, **xp_**, **ms_** and **sysmail_**.

Discovered CITs

To view discovered CITs, select a specific adapter in the Resources pane. For details, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

For details on the CIs that are discovered, see the section describing discovery progress and results in the *HP Universal CMDB Data Flow Management Guide*.

Note: To view the topology, see "Topology" on page 462.

SQL Server by OS Credentials Discovery

Universal Discovery can discover MS SQL Server CIs using operating system (OS) credentials. Universal Discovery creates an identifiable SQL Server CI, rather than a generic RunningSoftware CI.

Previously, SQL Server discovery assumed the existence of a process with the name of **sqlservr.exe**. Once Universal Discovery found this process, generic running software with a **MSSQL DB** value in the **name** attribute was reported to UCMDB.

The Data Flow Probe can report multiple SQL Server instances, each of them linked by a dependency link to its own **sqlservr.exe** process.

Universal Discovery supports SQL Server named instances.

There are two approaches to identifying MS SQL Server instance names by OS credentials. The changes appear in the **Host_Resources_Basic** package:

- **By Process Command Line**. The SQL Server process usually includes the MS SQL Server instance name in its command line. Universal Discovery extracts this instance name to a CI.

Note: A process command line cannot be retrieved by the SNMP protocol. Therefore, SNMP cannot be used to discover the MS SQL Server instance name, and Universal Discovery reports the generic running software CI instead.

- **Using Windows Services**. Universal Discovery checks existing services for those that include **sqlservr.exe** in the command line and extracts the instance name from the service name (because the service name reflects the instance name).

Chapter 34: SAP MaxDB Discovery

This chapter includes:

Overview

SAP MaxDB is an ANSI SQL-92 (entry level) compliant relational database management system (RDBMS) from SAP AG. The MaxDB discovery package provides shallow and deep discovery of MaxDB resources.

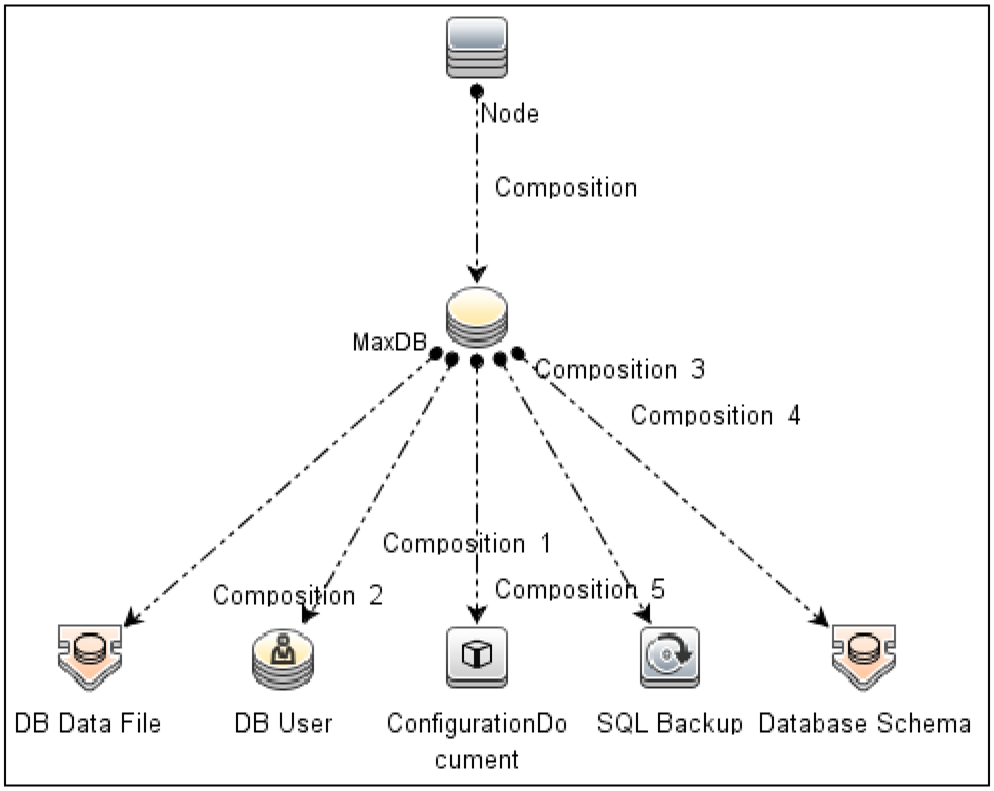
Supported Versions

This discovery supports SAP MaxDB 7.8.

Topology

The following image displays the topology of the SAP MaxDB Database discovery:

For a list of discovered CITs, see "Discovered CITs" on page 472.



How to Discover SAP MaxDB

This section describes how to discover the topology of SAP MaxDB.

This task includes the following steps:

1. **Prerequisite - Connectivity and user store**

   a. Shell connectivity to a MaxDB Node.

   b. Properly configured key store containing one key for each MaxDB instance being discovered.

   > **Note:** Because the command **xuser** is used to run the **dbmcli** tool, you must create a key store on the destination so the call for the tool is properly authenticated.

2. **Prerequisite - Set up protocol credentials**

   Define one of the following credentials, depending on the platform:

   - SSH
   - Telnet
   - NTCMD

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **Host Connection by Shell** job to discover the target host and shell connectivity to it.

   c. Run the **Host Applications by Shell** job to discover the resources of the target host, including MaxDB software and relevant processes.

   d. Run the **MaxDb by Shell** job to discover the topology of the target MaxDB database.

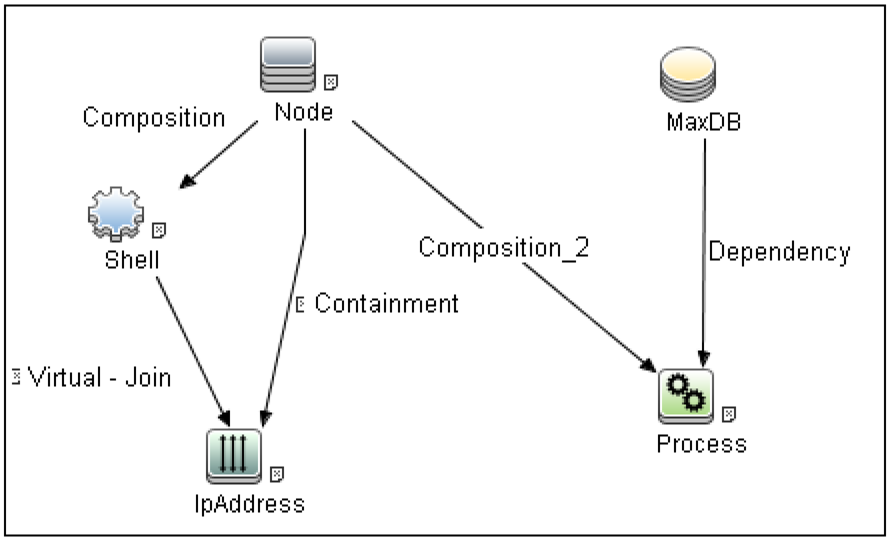
MaxDb by Shell Job

This section contains details about the job.

Adapter

This job uses the **MaxDb by Shell** adapter.

Trigger Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is Null
Process	(Name Equal kernel OR Name Equal kernel.exe) AND NOT Process Path Is null
Shell	NOT Reference to the credentials dictionary entry Is null

Parameters

None.

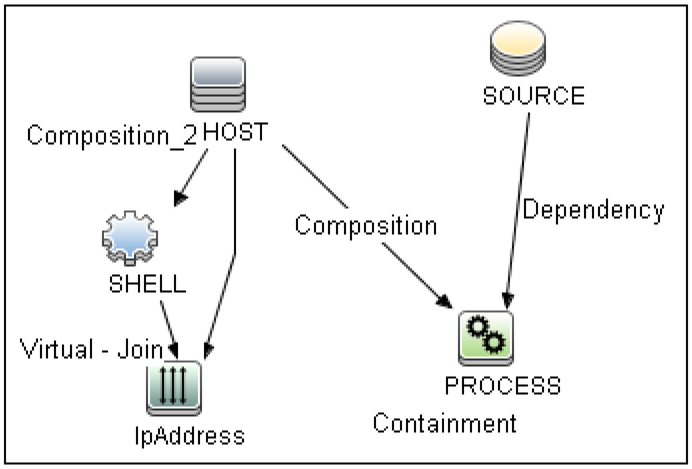
MaxDb by Shell Adapter

This section contains details about the adapter.

Input CIT

MaxDB

Input Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
PROCESS	Name Equal kernel OR Name Equal kernel.exe

Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
dbDataPath	${SOURCE.data_path}
dbPort	${SOURCE.application_port:}

Name	Value
dbProgramPath	${SOURCE.program_path}
dbSID	${SOURCE.name}
dbVersion	${SOURCE.application_version}
ip_address	${SHELL.application_ip}
processParams	${PROCESS.process_parameters:}
processPath	${PROCESS.process_path:}

Used Scripts

- entity.py
- db.py
- db_platform.py
- db_builder.py
- maxdb.py
- maxdb_discoverer.py
- maxdb_by_shell.py

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- DB Data FIle
- DB User
- Database Schema

- IpAddress
- IpServiceEndpoint
- MaxDB
- Node
- SQL Backup

Chapter 35: MySQL Replication Between Databases Discovery

This chapter includes:

Overview

This chapter explains how to discover MySQL database servers that replicate data in a master-slave relationship.

Replication enables data from one MySQL database server (the master) to be replicated to one or more MySQL database servers (the slaves). For details on replication, see the MySQL manual on the MySQL Web site: http://dev.mysql.com/doc/refman/5.0/en/replication-howto.html.

Currently all information about databases is retrieved through Shell protocols from the MySQL configuration file.

The job responsible for MySQL discovery is **MySQL by Shell**.

Supported Versions

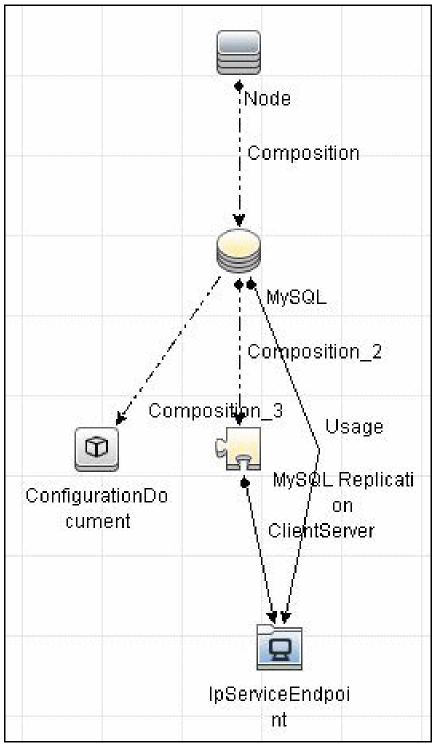
This discovery supports the following:

- MySQL versions 4.x, 5.x, 6.0
- Operating systems: Windows, Solaris, and Linux

Topology

Note: For a list of discovered CITs, see "MySQL by Shell Job" on page 479.

MySQL Replication Job



How to Discover MySQL Configuration and Replication Jobs

This task describes how to discover the MySQL configuration and replication jobs and includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the following protocols:

   - SSH Protocol
   - Telnet Protocol
   - NTCMD Protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Retrieve information**

   To retrieve all relevant information, DFM must have read permissions for the $MYSQL_HOME directory and for executing **mysqld** (**mysqld.exe** or **mysqld-nt.exe** for Windows) with the following parameters:

   ```
   mysqld --verbose --help
   ```

   ```
   mysqld --version
   ```

   If the **my.cnf** (**my.ini**) file is located outside the `$MYSQL_HOME` directory, you must add permissions for reading to it.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up and running.

   b. Run the **Host Connection by Shell** job to create Shell CITs.

   c. Run any of host resources jobs to gather information about processes running on the host.

   d. Run the **MySQL by Shell** job to retrieve information about MySQL configuration and replication jobs.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

MySQL by Shell Job

This section includes details about the jobs.

Discovery Mechanism

This section explains how DFM discovers the MySQL server:

1. The MySQL by Shell job connects to the remote host using Shell credentials.
2. The job checks for the existence of the path of the MySQL configuration file by executing the following command:

   ```
   mysqld --verbose --help
   ```

3. If the job cannot find the configuration file with this command, it assumes the file is located in the default configuration file path:
   - UNIX or Linux: **/etc/my.cnf**
   - Windows: **../my.ini**
4. The job tries to retrieve the attribute values from the configuration file. The job either reads the attribute values from the command line, or reads the configuration file to find the values of the attributes that were not found in the command line.

   Example of command line with attribute values:

   ```
   mysqld-nt.exe --defaults-file=C:\hp\UCMDB\DataFlowProbe\MySQL\my.ini DDM_Probe_
   DB
   ```

5. If the job does not find any attribute values, it takes the default values from the MySQL documentation.

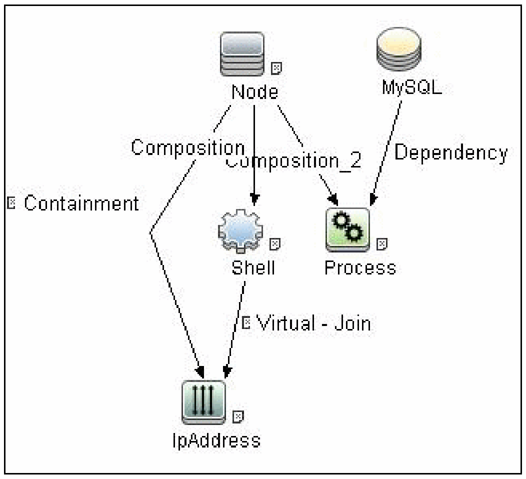
   For details of the MySQL attributes, see "MySQL by Shell Job" above.

6. The job creates the MySQL CIs with appropriate attribute values and relationships.
7. The job now checks if this MySQL instance is a replica. If it is a replica, the job attempts to discover a master host and master user. The version of the MySQL engine is taken from the **mysqld --**

version command output.

8. The job creates the MySQL replication CI with appropriate attribute values and relationships.

Trigger Query



Configuration Item Types

Name	Parent CIT	Uses Existing Attributes	Uses New Attributes	Description
MySQL	Database	database_ dbsid	server_id, database_datadir, database_max_ connections	CIT represents the MySQL database
MySQL Replication	DB Scheduler Job		master_user, master_ connect_ retry	CIT represents the MySQL Replication job

CIT Attributes

- MySQL
  - server_id. The server ID is used in the replication job and must be unique for each server.
  - database_datadir. Path to the database root (datadir in the configuration file).
  - database_max_connections. The maximum number of concurrent sessions allowed by the MySQL

server (max_connections in the my.ini file).

- database_dbsid. The unique identifier for running the MySQL instance-process port. The format is MySQL on port ####.

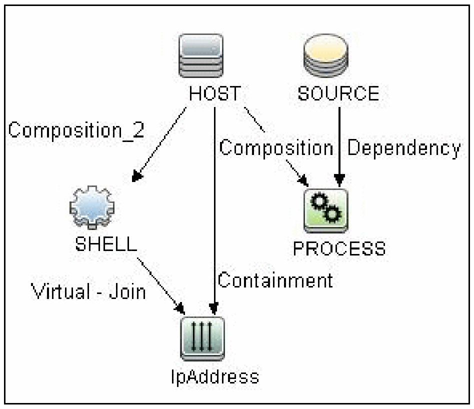
- MySQL Replication
    - master_user. A user name used when connecting to the master server.
    - master_connect_retry. The number of seconds that the slave thread sleeps before trying to reconnect to the master, if the master goes down or the connection is lost.

Relationships

Source	Destination	Relationship Type	Cardin-ality
mysql	configfile	Composition	1..1
mysql	mysql_replication	Composition	1..1
mysql_replication	IpServiceEndpoint	ClientServer	1..1

Adapter

- Input Query

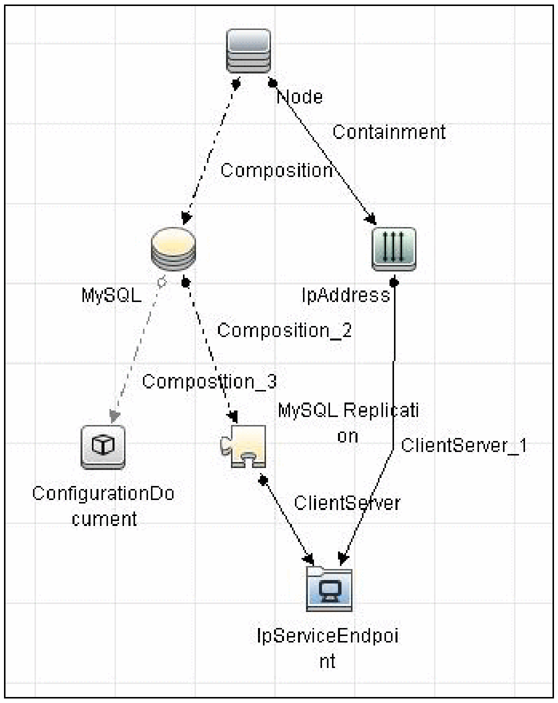


- Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
dbport	${SOURCE.database_dbport}
dbsid	${SOURCE.database_dbsid}
ip_address	${SHELL.application_ip}
processParams	${PROCESS.process_parameters}
processPath	${PROCESS.process_path}

Discovered CITs

To view discovered CITs, select a specific adapter in the Resources pane.



- ClientServer
- Composition

- ConfigurationDocument
- Containment
- IpAddress
- IpServiceEndpoint
- MySQL
- MySQL Replication
- Node

Note: To view the topology, see "Topology" on page 476.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for MySQL Replication Between Databases discovery.

- There are two main approaches to running several active MySQL instances on one host:
  - Two MySQL instances are each run on a different port, for example, one on 134.44.1.1:3306 and the second on 134.44.1.1:3307.
  - A host has several IPs, and each MySQL process is bound to its own IP, for example, 134.44.1.1:3306 and 134.44.1.2:3306.

  In the second case, as the key identifier that differentiates one MySQL CI from another is a port number (without an IP), the job cannot differentiate between the two MySQL instances and merges them into one CI.

Chapter 36: Oracle Database Server Discovery

This chapter includes:

Overview

There are two types of Oracle Database Server discoveries:

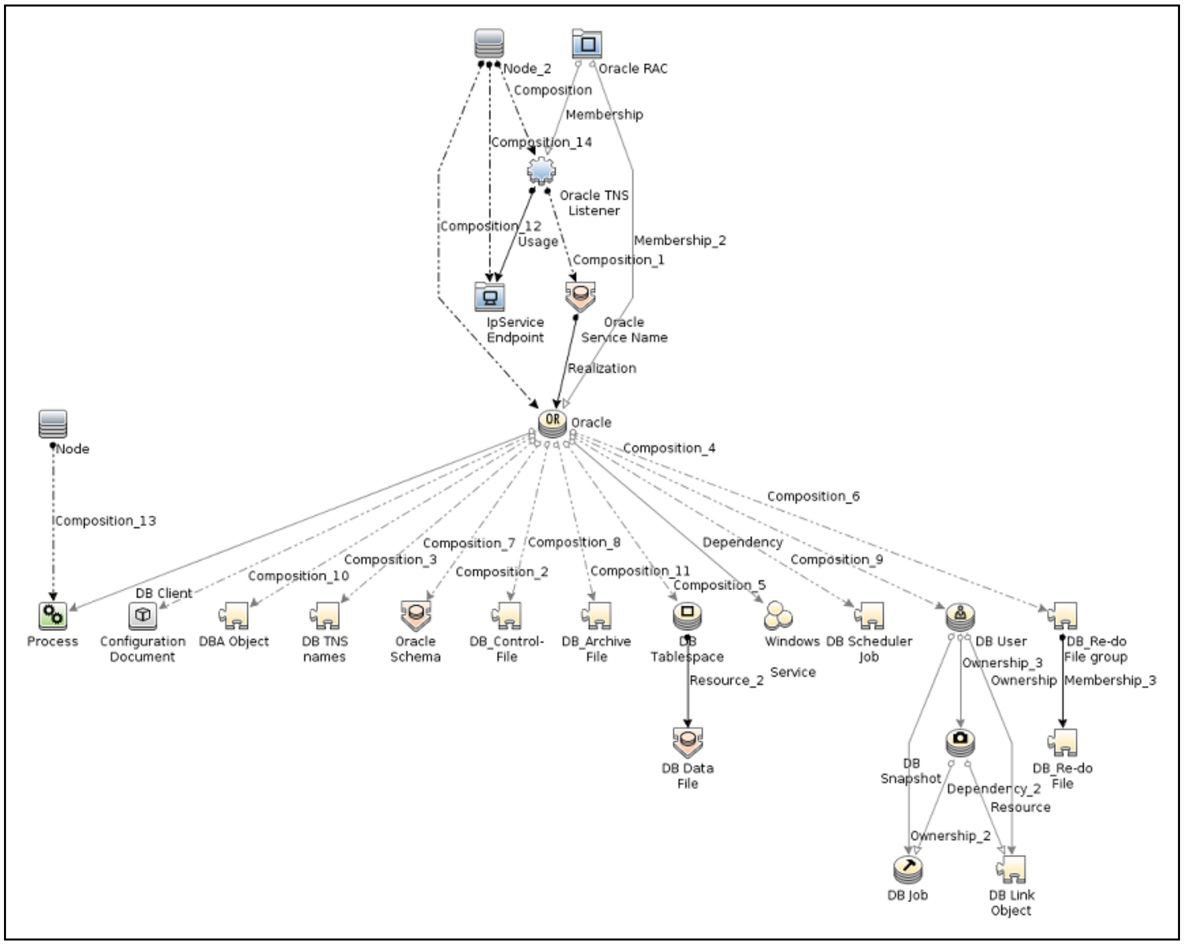
- The general Oracle Database Server discovery, which discovers all the Oracle Database Servers on the network. There are two methods available:
  - **Thorough Discovery**. This is a comprehensive discovery method that uses Node CI as an input trigger CIT. For information on this type of discovery, see "How to Discover Oracle Database Servers" on page 487.
  - **Lightweight Discovery**. This is a lightweight method of discovery that uses IPServiceEndpoint as an input trigger CIT. For information on this type of discovery, see "How to Discover Oracle Database Servers- Lightweight" on page 489.
- Oracle TNS Names Topology discovery, which discovers all the Oracle Database Servers that are stored in the Active Directory using the LDAP protocol. For more information on this type of discovery, see "How to Discover Oracle TNS Names Topology" on page 488.

Supported Versions

Oracle Database Server discovery and Oracle TNS Names Topology discovery support Oracle 8, 9, 9i, 10, 10g, 11g, and 12c.

Topology

The following image displays the topology of the Oracle Database Server discovery:


Node_2
Oracle RAC
Composition
Membership
Composition_14
Oracle TNS Listener
Composition_12
Usage
Membership_2
Composition_1
IpService Endpoint
Oracle Service Name
Realization
Oracle
Node
Composition_13
Composition_4
Composition_6
Composition_7
Composition_8
Dependency
Composition_9
Composition_3
Composition_10
Composition_2
Composition_11
Composition_5
DB Client
Process
Configuration Document
DBA Object
DB TNS names
Oracle Schema
DB_Control-File
DB_Archive File
DB Tablespace
Windows Service
DB Scheduler Job
DB User
DB_Re-do File group
Resource_2
Ownership_3
Ownership
Membership_3
DB Data File
DB Snapshot
DB_Re-do File
Dependency_2
Resource
Ownership_2
DB Job
DB Link Object

How to Discover Oracle Database Servers

This task describes how to discover Oracle databases. This discovery adds a valid credentials ID to the CMDB. You can then use this CI to fully discover the database.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Oracle Database Server discovery uses the Generic DB Protocol (SQL).

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Verify user on Oracle database server**

   Run the **Databases TCP Ports** job to verify the user name, password, and port used by the Oracle Database Server.

3. **Run the discovery**

   Activate the following jobs in the listed order:

   a. **Databases TCP Ports**

   b. **Oracle Database Connection by SQL**

   c. **Oracle Topology by SQL**

      Note: Due to the large amount of data reported by the Oracle Topology by SQL job, topology data is sent in chunks. Chunk size (the number of objects in a chunk) is regulated by the **discoverReportPageSize** job parameter. The default value is 1,000 objects in one chunk.

   For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Oracle TNS Names Topology

This section describes how to discover Oracle TNS Names Topology from the Active Directory using the LDAP protocol.

Prerequisites

- Ensure that an Active Directory Server has been properly configured, on which Oracle TNS information is stored.
- Ensure that LDAP protocol credentials have been properly configured.

Run the discovery

1. Run the **Range IPs by ICMP** job to discover the target IPs.
2. Run the **TCP Ports** job to discover LDAP ports. Ensure that the LDAP port was discovered.
3. Run the **Active Directory Connection by LDAP** job to discover the Active Directory servers.
4. Run the **Oracle TNS Names by LDAP** job to discover the Oracle TNS Names topology.

How to Discover Oracle Database Servers- Lightweight

This task describes how to discover Oracle databases using a lightweight method of discovery. This discovery adds a valid credentials ID to the CMDB. You can then use this CI to fully discover the database.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Oracle Database Server discovery uses the Generic DB Protocol (SQL).

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Verify user on Oracle database server**

   Run the **Databases TCP Ports** job to verify the user name, password, and port used by the Oracle Database Server.

3. **Run the discovery**

   Activate the following jobs in the listed order:

   a. **Databases TCP Ports**

   b. **Oracle Database Connection by SQL- Lightweight**

   c. **Oracle Topology by SQL**

      Note: Due to the large amount of data reported by the Oracle Topology by SQL job, topology data is sent in chunks. Chunk size (the number of objects in a chunk) is regulated by the **discoverReportPageSize** job parameter. The default value is 1,000 objects in one chunk.

   For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

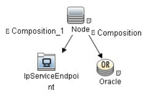
Oracle Database Connection by SQL- Lightweight Job

Adapter

ID: `SQL_NET_Dis_Connection_Oracle_Lightweight`

Trigger TQL

- Trigger CI: IpServiceEndpoint
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
Source	IpServiceName Equal ignore case oracle OR ServiceNames Contains oracle

Discovery Flow

This job performs the following actions:

1. Gets the Oracle credentials that are specified in the Discover Probe configuration.
2. Tries to connect using the specific credentials.
3. If connection is successful, reports the Oracle Database with the connected credentials.

Oracle Connection by SQL- Lightweight Adapter

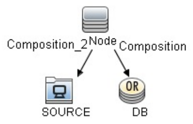
ID

Oracle Connection By SQL- Lightweight

Input CIT

IpServiceEndpoint

Input TQL



Triggered CI Data

Name	Value
application_ip	${DB.application_ip:NA}
application_port	${DB.application_port:NA}
sa_ip	${SA.bound_to_ip_address:NA}
sa_port	${SA.network_port_number:NA}
sid	${DB.name:NA}

Used Scripts

- file_ver_lib.py
- SQL_Connection.py

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node
- Oracle
- Usage

Parameters

- **protocolType.** Default value is **oracle**, should not be changed.

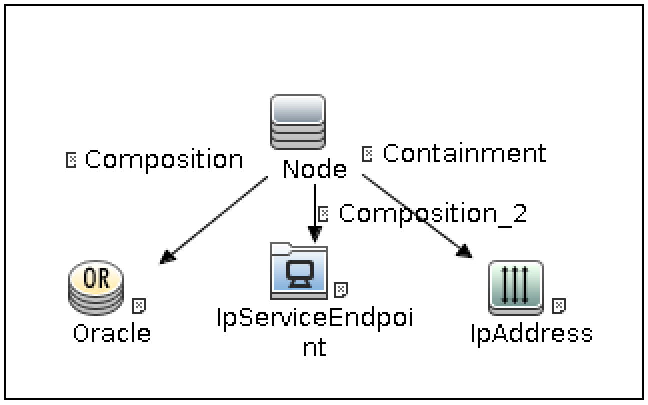
Oracle Database Connection by SQL Job

Adapter

ID: `Oracle Connection By SQL`

Trigger TQL

- Trigger CI: Node
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
IpAddress	NOT IP Probe Name Is null
IpServiceEndpoint	Name Equal ignore case "oracle"

Discovery Flow

This job performs the following actions:

1. Gets the Oracle credentials that are specified in the Discover Probe configuration.
2. Tries to connect using the specific credentials.
3. If connection is successful, reports the Oracle Database with the connected credentials.
4. If connection is successful, asks the database for it's primary IP and checks if this IP matches the one used for connection.

   If the primary IP does not match the one used for connection, the job checks if the database is in clustered mode.
5. If the IP reported by the database is reachable from the probe and it is possible to connect to it, that IP is populated to the UCMDB. Otherwise, the IP used for initial connection is populated to the UCMDB.

Oracle Connection by SQL Adapter

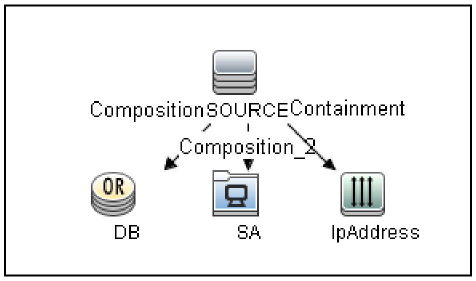
ID

Oracle Connection By SQL

Input CIT

Node

Input TQL



Triggered CI Data

Name	Value
application_ip	${DB.application_ip:NA}
application_port	${DB.application_port:NA}
ip_address	${IpAddress.name}
sa_ip	${SA.bound_to_ip_address:NA}
sa_port	${SA.network_port_number:NA}
sid	${DB.name:NA}

Used Scripts

- file_ver_lib.py
- SQL_Connection.py

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node
- Oracle
- Usage

Parameters

- **protocolType.** Default value is **oracle**, should not be changed.

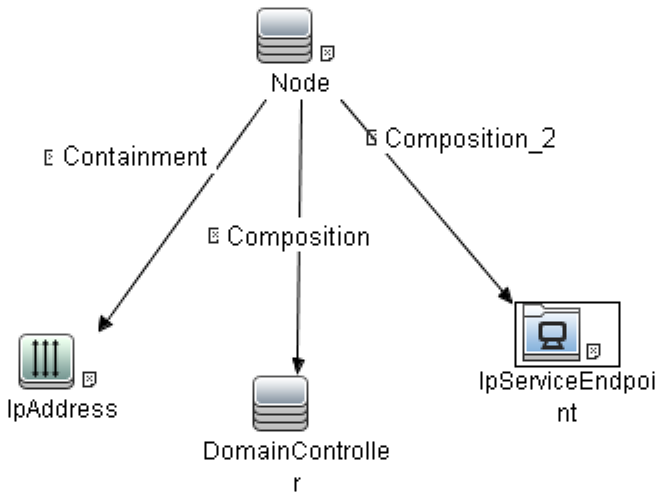
Oracle TNS Names by LDAP Job

Adapter

ID: `oracle_tns_names_by_ldap`

Trigger TQL

- Trigger CI: IpAddress
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
Source	NOT IP Probe Name Is null
IpServiceEndpoint	Name Equal ignore case "ldap"

Parameters

By default, parameters are not overridden and use values from the adapter.

Discovery Flow

1. Get **DomainContext** from the root (if the **baseDN** parameter is not specified).

2. Get the **orclContext** object from the previous node.

3. Get the **orclNetService** object from the **orclContext** node.

4. Get the **orclNetDescString** attribute from the **orclNetService** object.

Oracle TNS Names by LDAP Adapter

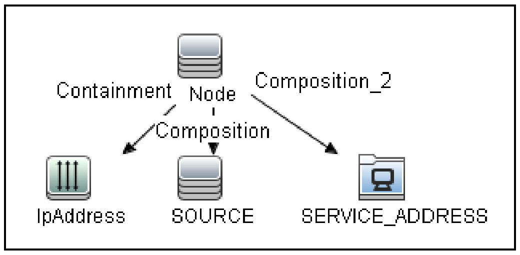
ID

oracle_tns_names_by_ldap

Input CIT

DomainController

Input TQL



Triggered CI Data

Name	Value
application_port	${SOURCE.application_port:NA}
credentials_id	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
port	${SERVICE_ADDRESS.network_port_number}

Used Scripts

- active_directory_utils.py
- db.py
- db_builder.py
- db_platform.py

- jdbc_url_parser.py
- oracle_by_ldap.py
- oracle_ldap_discoverer.py

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node
- Oracle
- Oracle Service Name
- Oracle TNS Listener
- Realization
- Usage

Parameters

- **baseDN.** Specifies the context in which to find the **orclContext** objects.

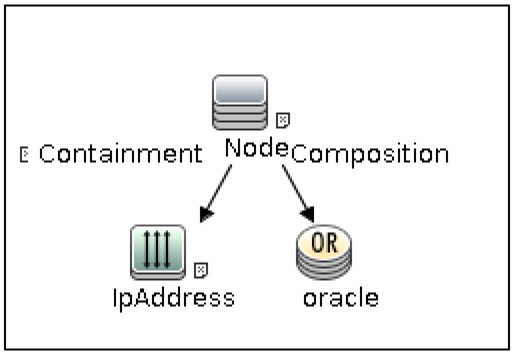
Oracle Topology by SQL Job

Adapter

ID: `Oracle database topology by SQL`

Trigger TQL

- Trigger CI: Oracle
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
IpAddress	NOT IP Probe Name Is null
oracle	NOT Reference to the credentials dictionary entry Is null

Discovery Flow

1. Connect to the Oracle Database using credentials ID that is specified in the Oracle CI.
2. Execute the specific SQL queries, according to the flags specified in the adapter parameters.
3. Push the discovered data to the UCMDB.

Oracle Database Topology by SQL Adapter

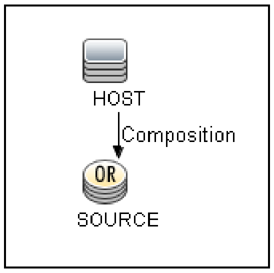
ID

`Oracle database topology by SQL`

Input CIT

Oracle

Input TQL



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
port	${SOURCE.application_port:NA}
sid	${SOURCE.name:NA}

Used Scripts

- SQL_Dis_Oracle.py

Discovered CITs

- Composition
- Containment
- DB Client
- DB Job
- DB Link Object

- DB Scheduler Job
- DB Snapshot
- DB Tablespace
- DB User
- DB_Archive File
- DB_Control-File
- DB_Re-do File group
- DB_Re-do File
- DBA Object
- Dependency
- IpAddress
- Membership
- Node
- Oracle RAC
- Oracle Schema
- Oracle
- Ownership
- Process
- Resource

Parameters

- **comprehensiveDiscovery.** If false, the DBA Object, DB Job and DB User are not retrieved. The default value is false. If the parameter is set to false, changing other parameters has no effect on the discovery process (DBA Objects are not discovered).

- **discoverFunctions.** If false, functions are not retrieved. The default value is false.
- **discoverPackageBody.** If false, package bodies are not retrieved. The default value is false.
- **discoverPackages.** If false, packages are not retrieved. The default value is false.
- **discoverProcedures.** If false, procedures are not retrieved. The default value is false.
- **discoverTables.** If false, tables are not retrieved. The default value is false.
- **discoverReportPageSize.** The maximum amount of query result objects that can be sent in one bulk. The default value is 1,000.
- **discoveryUsers.** If true, database users are retrieved. If false, database user discovery is controlled by the **comprehensiveDiscovery** parameter.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Oracle Database Server discovery.

- If you need to discover Oracle 8g, use the following values in **jdbcDrivers** and **jdbcPreUrls**:

  jdbcDrivers:

  ```
  <oracle>com.inet.ora.OraDriver</oracle>
  <oracleSSL>com.mercury.jdbc.oracle.OracleDriver</oracleSSL>
  ```

  jdbcPreUrls:

  ```
  <oracle>jdbc:inetora:%%ipaddress%%:%%protocol_port%%:%%sqlprotocol_
  dbsid%%?logging=false&amp;loginTimeout=%%protocol_timeout%%</oracle>
  <oracleSSL>jdbc:mercury:oracle://%%ipaddress%%:%%protocol_
  port%%;ServiceName=%%sqlprotocol_dbsid%%</oracleSSL>
  ```

Chapter 37: Oracle Real Application Cluster (RAC) Discovery

This chapter includes:

Overview

DFM discovers information about Oracle RAC through the Shell protocols from the Oracle configuration files **listener.ora** and **tnsnames.ora**, and through the **lsnrct** utility.

Supported Versions

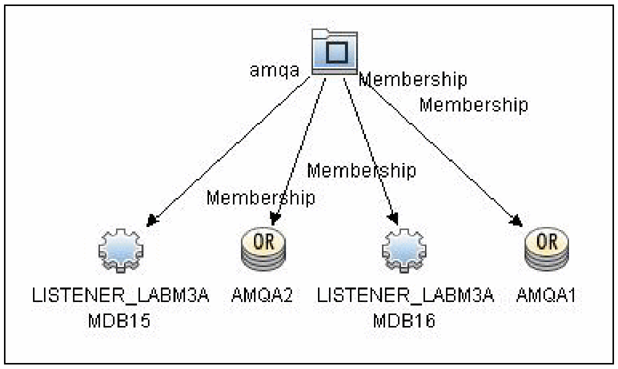
This discovery supports Oracle DB 10g, 11g, and 12c.

Topology

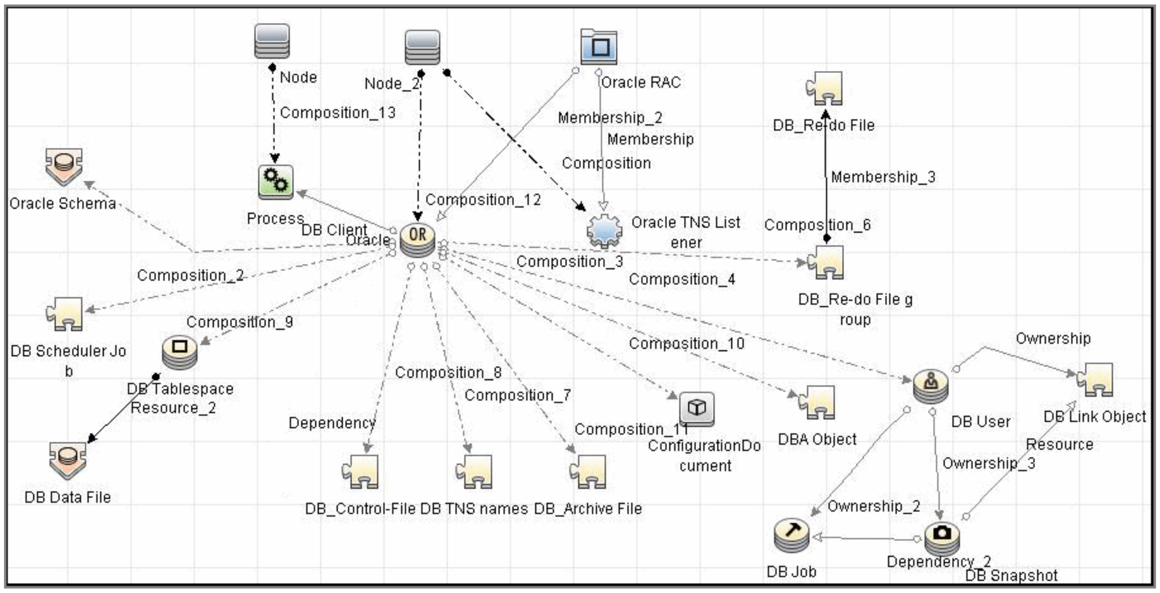
The following images display sample output of the Oracle RAC discovery topology.

Note: For a list of discovered CITs, see "Oracle Listeners by Shell Job" on page 506 and "Oracle RAC Topology by Shell Job" on page 510.

- **Topology**



- **Oracle View**



How to Discover Oracle Real Application Cluster (RAC)

This section includes the following topics:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the NTCMD, SSH, or Telnet protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   a. To retrieve all relevant information, verify that DFM has:

      - Read permissions for the **$ORACLE_HOME\network\admin** directory

      - The correct execute permissions for **$ORACLE_HOME\bin\lsnrctl** and for the corresponding library (lib) and message files.

   b. **Oracle Listeners by Shell job**. Verify that the RAC relative processes are running on the Oracle database. The file names begin with **ora_lms**, **ora_lmd**, **ora_lck**, and **oracm**.

c. **Oracle RAC Topology by Shell job**. The **Listened IPs** of the Listener CIT must be **not NULL**.

d. Run the **Host Connection by Shell** job, to activate Shell CITs.

3. **Run the discovery**

   a. Run any of the host resources jobs that gather information about processes running on the host. For example, **Host Applications by Shell**.

   If DFM discovers TNS Listener processes, the job creates Oracle TNS Listener CIs and an Oracle DB CI together with its connected processes.

   b. To discover Oracle TNS Listener CIs with full data, run the **Oracle Listeners by Shell** job. This job connects to the host and retrieves the required data for the Oracle TNS Listener CI. For details, see "Oracle Listeners by Shell Job" below.

   c. To discover Oracle RAC topology, run the **Oracle RAC Topology by Shell** job. This job connects to the hosts with full listeners and discovers RAC. For details, see "Oracle RAC Topology by Shell Job" on page 510. For details on undiscovered elements, see "Troubleshooting and Limitations" on page 515.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Oracle Listeners by Shell Job

This section includes:

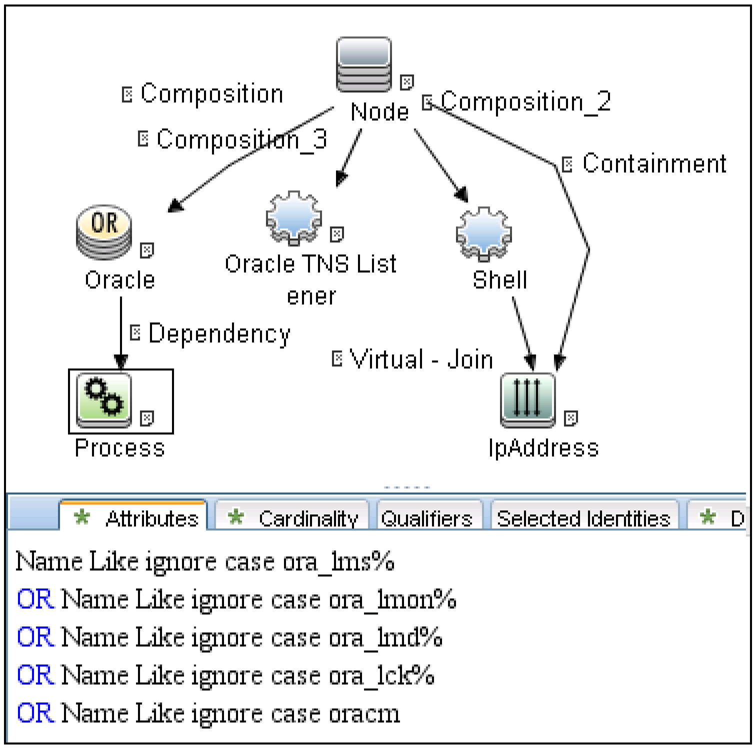
- "Oracle Listeners by Shell Job" above
- "Oracle Listeners by Shell Job" above
- "Oracle Listeners by Shell Job" above
- "Oracle Listeners by Shell Job" above

Discovery Mechanism

This job triggers on Oracle databases that have RAC related processes. The job:

1. Connects to the remote host by Shell.
2. Checks for the **ORACLE_HOME** environment variable.
3. If the variable is not defined, the job takes the **ORACLE_HOME** value from the job adapter (if defined).
4. Reads the **Oracle TNS listener** configuration file, stored in **$ORACLE_HOME/network/admin/listener.ora**, and performs further parsing.
5. Retrieves a full list of IP addresses to which this particular listener is listening.
6. Checks for listener status using the **$ORACLE_HOME/bin/lsnrctl** status.
7. Retrieves known services and listener status from the output.

Trigger Query

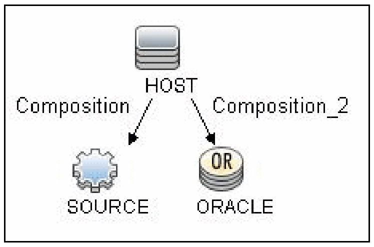


Name Like ignore case ora_lms%
OR Name Like ignore case ora_lmon%
OR Name Like ignore case ora_lmd%
OR Name Like ignore case ora_lck%
OR Name Like ignore case oracm

Adapter

This job uses the **Oracle_Listeners_by_Shell** adapter.

- **Input Query**



- **Used Scripts**

  oracle_listeners_by_shell.py

- **Triggered CI Data**

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
ip_address	${SHELL.application_ip}

- **Adapter Parameters**

OracleHomes	Used when no **ORACLE_HOME** environment variable is defined. This value must be the same as the parameter in the Oracle RAC Topology by Shell job.

Discovered CITs

- Composition
- Containment
- IpAddress
- Node
- Oracle TNS Listener
- Unix

Note: To view the topology, see "Topology" on page 504.

Oracle RAC Topology by Shell Job

This section includes:

- "Oracle RAC Topology by Shell Job" above
- "Oracle RAC Topology by Shell Job" above
- "Oracle RAC Topology by Shell Job" above
- "Oracle RAC Topology by Shell Job" above

Discovery Mechanism

This job:

1. Connects to the remote host by Shell.
2. Checks for the **ORACLE_HOME** environment variable.
3. If it is not defined, the job uses the **OracleHome** value from the job adapter.
4. Enumerates configured database service names using the command **srvctl config database**.
5. Enumerates **sids** and hosts on which the service is available using the command **srvctl status database –d <service_name>**.
6. Retrieves RAC parameters such as Service Name and Nodes from the **$ORACLE_HOME/network/admin/tnsnames.ora** file.
7. Checks if this RAC instance is running, by parsing the **lsnrctl status** output.

   Note: Nodes are cited in the **tnsnames.ora** file by their internal IP or by their internal domain name. If the domain name appears, DFM resolves it.

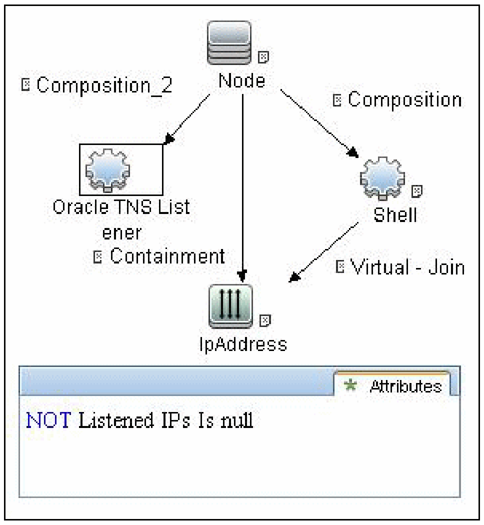
8. Retrieves the full list of Listened IPs from the input query, for all listeners matching the query.
9. Parses this attribute's values from the list of listened IPs, to retrieve the Host Primary Domain name that corresponds to the MAC address. This is needed since the RAC CI's name key attribute must consist of a list of all the node domain names separated by the colon symbol (:).
10. Looks up the full node name in the build table sorted by IP address. The result is the Host Primary

Domain name for each node.

At this stage, the following information is available: the RAC Service Name, the fully qualified domain names of all the RAC nodes, and a RAC instances count.

11. Creates the RAC CI.

Trigger Query

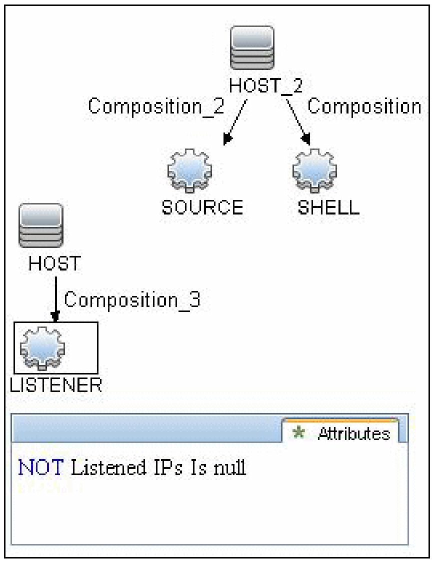


NOT Listened IPs Is null

Adapter

This job uses the **Oracle_RAC_Topology_by_Shell** adapter.

- **Input Query**



- **Triggered CI Data**

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
ip_address	${SHELL.application_ip}
listened_ips	${LISTENER.listened_ips}

- **Adapter Parameters**

OracleHomes	Used when no **ORACLE_HOME** environment variable is defined. This value must be the same as the parameter in the Oracle Listeners by Shell job.

Discovered CITs

- Composition
- Containment
- IpAddress

- Membership
- Node
- Oracle
- Oracle RAC
- Oracle TNS Listener
- Running Software

Note: To view the topology, see "Topology" on page 504.

Configuration Items

CI	Description
Oracle TNS Listener	This CIT represents the Oracle TNS Listener.
CIT name	oracle_listener
Parent CIT name	application
Key attributes	• **name (displayed as Name)**. The `TNS Listener` constant. • **root_container (displayed as Container)**. The Container CI. • **listener_name (displayed as Name of the Listener)**. The real `TNS Listener` name.
Additional Attributes	**listened_ips (displayed as Listened IPs)**. Listened to IP addresses and machine domain name. Listened IPs are IP addresses that are listened to by the Oracle TNS Listener. Format: `<host_name>:<host_primary_ip>@<listened_ip>:<mac>;... <listened_ip>:<mac>` **Note:** MAC addresses are not currently discovered. The marker acts as a placeholder for future enhancements.

Relationships

CIT	Link Type	Cardinality
Node	Composition	1.*
RAC	Membership	1.*
Process	Dependency	1.*

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Oracle RAC Discovery.

Error Message	**Description**
Failed to lookup host name. No RAC CI will be created.	For one or more nodes, the job failed to retrieve the FQDN (fully qualified domain name) from the listeners **listened_ips** attribute information. • Check the logs to retrieve the IP and destination. • Make sure that the FQDN for that IP can be obtained either from the DNS or from the host file.
No RAC CI are retrieved.	Not all nodes were discovered with the correct listener information.
Discovery cannot discover links to the remote machines (database clients)	This can occur in the following situation: The discovered database reports its clients by their host names and not by their IP addresses, and the host name cannot be resolved to an IP address. In this case, the remote client cannot be created.

Part 7: Enterprise Applications

Chapter 38: Active Directory Discovery

This chapter includes:

Overview

Active Directory (AD) provides an extensible and scalable directory service that enables efficient managing of network resources.

Data Flow Management discovers Active Directory topology through the LDAP Directory Service Interface that communicates with the AD domain controllers. Data Flow Management uses JNDI to provide the API that interacts with the LDAP Directory Service Interface.

Supported Versions

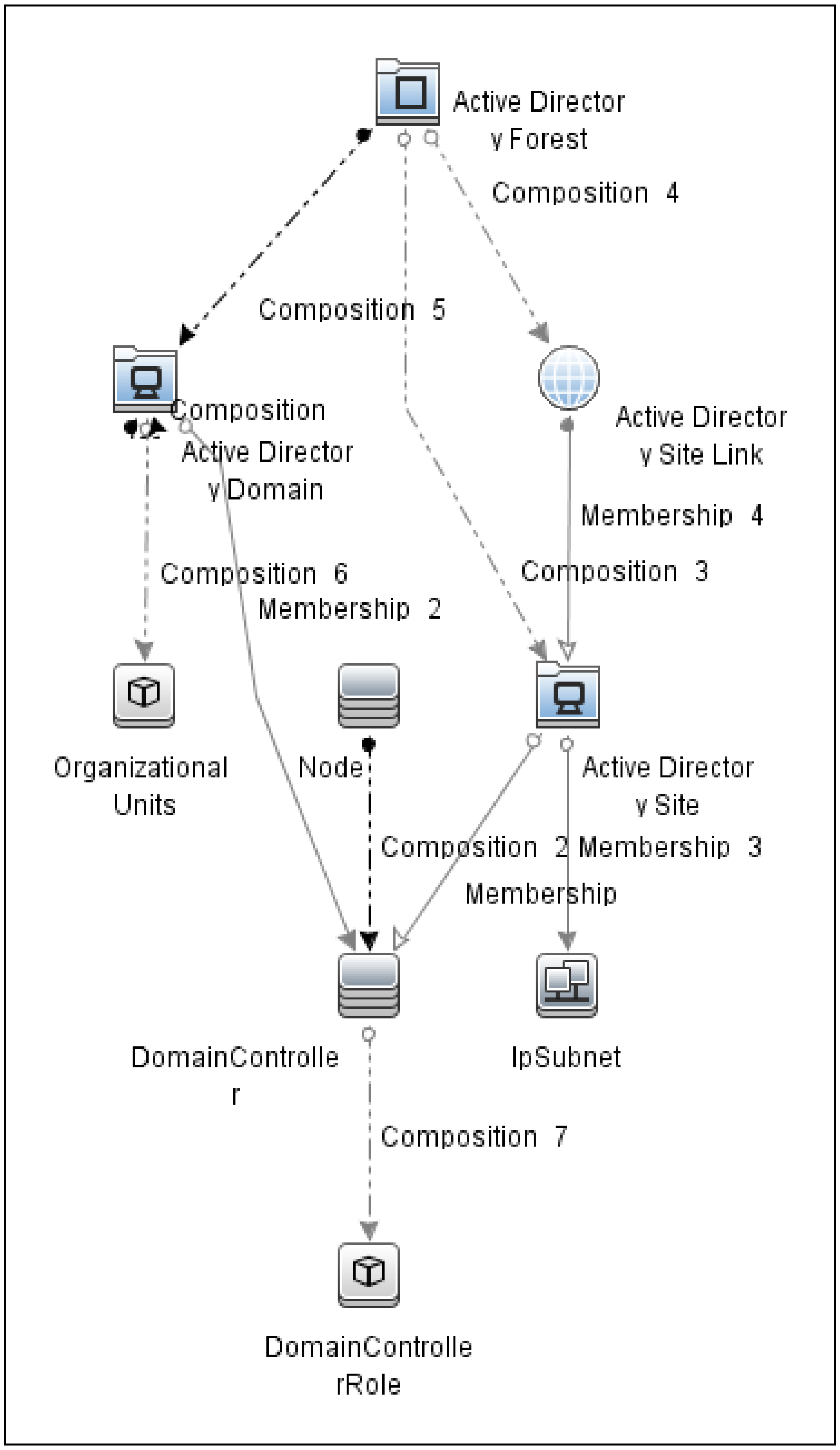
This discovery solution supports the following servers:

- Windows Server 2000
- Windows Server 2003
- Windows Server 2008

Topology

The following image displays the AD topology.

Note: For a list of discovered CITs, see "Discovered CITs" on page 527.


Active Director
y Forest
Composition 4
Composition 5
Composition
Active Director
y Domain
Active Director
y Site Link
Membership 4
Composition 6
Membership 2
Composition 3
Organizational
Units
Node
Active Director
y Site
Composition 2
Membership 3
Membership
DomainControlle
r
IpSubnet
Composition 7
DomainControlle
rRole

How to Discover Active Directory Domain Controllers and Topology

This task explains how to discover Active Directory and includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   a. To discover hosts, you must set up the SNMP, Shell (NTCMD, SSH, Telnet), and WMI protocols.

      - SNMP protocol

        Prepare the following information for the SNMP protocol: **community name** (for v2 protocol), **user name** (for v3 protocol), and **password** (for v3 protocol).

      - Shell Protocols: NTCMD, SSH, Telnet protocols

        Prepare the following information for the Shell protocol: **user name**, **password**, and **domain name** (optional for NTCMD).

      - WMI protocols

        Prepare the following information for the WMI protocol: **user name**, **password**, and **domain name** (optional).

   b. To run all AD jobs, you must set up the LDAP protocol. There are two versions of the protocol available: **2** and **3**. There is no formal standardization of version 2, therefore Data Flow Management uses the version 3 protocol.

      Note: User Name: if a domain is present, use **username@domain**.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Other**

   a. Discover the host of each AD domain controller: activate one of the following jobs (depending on the protocol you are using):

- **Host Connection by Shell**
- **Host Connection by SNMP**
- **Host Connection by WMI**

b. Verify that the **portNumberToPortName.xml** configuration file includes all possible AD ports. For example, if AD is running on LDAP port 389, locate the following row in the file:

```
<portInfo portProtocol="tcp" portNumber="389" portName="ldap" discover="0"
/>
```

Change the **discover="0"** attribute value to **discover="1"**.

For details, see the sections about the portNumberToPortName.xml file, and about a New Port, in the *HP UCMDB Universal Discovery Content Guide - General Reference* document.

c. To discover all known LDAP ports, perform the following additional configuration to the TCP Ports job:

- set **ports** option to **ldap**

d. Open the LDAP port of the destination IP for each domain controller server by activating the following job in the **Tools and Samples > Discovery Tools** module:

- **TCP Ports.** This job includes the **TCP_NET_Dis_Port** adapter.

3. **Run the discovery**

- Activate the **Active Directory Connection by LDAP** job. This job discovers the existence of AD domain controllers through LDAP. For query and parameter details, see "Active Directory Connection by LDAP Job" below.

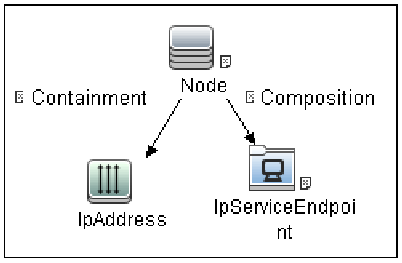
- Activate the **Active Directory Topology by LDAP** job. This job connects to the AD domain controller servers and discovers their topology. For query and parameter details, see "Active Directory Topology by LDAP Job" on page 525.

Active Directory Connection by LDAP Job

This section contains details about the job.

Trigger Query

- Trigger CI: IpAddress
- Trigger query:



- CI attribute conditions:

CI	Attribute Value
Source	NOT IP Probe Name Is null
IpServiceEndpoint	Name Equal ignore case "ldap"

Adapter

This job uses the **LDAP_Active_Directory_Connection** adapter.

- Triggered CI Data

Name	Value	Description
hostId	${HOST.root_id}	The ID of the host on which the domain controller resides.
ip_address	${SOURCE.ip_address}	The IP address, retrieved from the IpServiceEndpoint.
port_number	${Service_Address. ipport_number}	The LDAP port number, retrieved from the IpServiceEndpoint.

- Adapter Parameters

Parameter	Description
baseDn	This is the domain name where records about domain controllers are stored. **Default**: OU=Domain Controllers

Discovered CITs

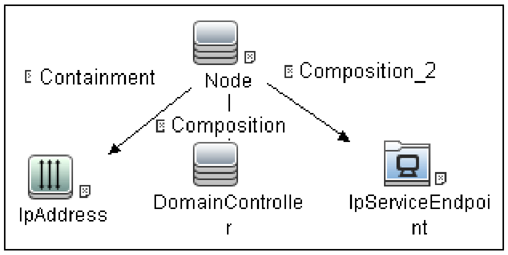
- Containment
- Composition
- DomainController
- Node
- IpAddress

Active Directory Topology by LDAP Job

This section contains details about the job.

Trigger Query

- Trigger CI: DomainController
- Trigger Query:



- CI attribute conditions:

CI	Attribute Value
IpAddress	NOT IP Probe Name is null
Source	▪ NOT Reference to the credentials dictionary entry Is null ▪ NOT Application IP is null
IpServiceEndpoint	`Name Equal ignore case "ldap"`

Adapter

This job uses the **LDAP_Active_Directory_Topology** adapter.

- Triggered CI Data

Name	Value	Description
application_ port	${SOURCE.application _ port:NA}	The port retrieved from the IpServiceEndpoint.
credentials_id	${SOURCE.credentials _id}	The credentials ID of the protocol saved in the domain controller's attribute.
hostId	${HOST.root_id}	The ID of the host on which the domain controller resides.
ip_address	${SOURCE.application_ip}	The IP address of the server.
port	${SERVICE_ ADDRESS.network_port_ number}	The LDAP port number.

- Adapter Parameters

Parameter	Description
tryToDiscoverGlobalCatalog	If this parameter is set to **true**, DFM attempts to discover the entire topology by connecting to the domain controller designated as a global catalog server. The connection is made through the port defined in the **globalCatalogPort** parameter. **Default**: true - the global catalog is used for discovery
globalCatalogPort	The port number through which DFM accesses the domain controller designated as the global catalog. **Default: 3268** **Note**: This parameter is needed only when **tryToDiscoverGlobalCatalog** is set to true.
baseDn	This is the domain name where records about domain controllers are stored. **Default**: OU=Domain Controllers

Discovered CITs

- Active Directory Domain. Domains in the AD Forest.
- Active Directory Forest. Information about functionality level and contiguous names.
- Active Directory Site. Available site objects that are configured in the AD Forest.
- Active Directory Site Link
- Active Directory System
- Composition
- Containment
- ConfigurationDocument
- DomainController
- DomainControllerRole
- Node
- Membership. Relationships between sites and subnets.
- IpSubnet. Available subnet objects.

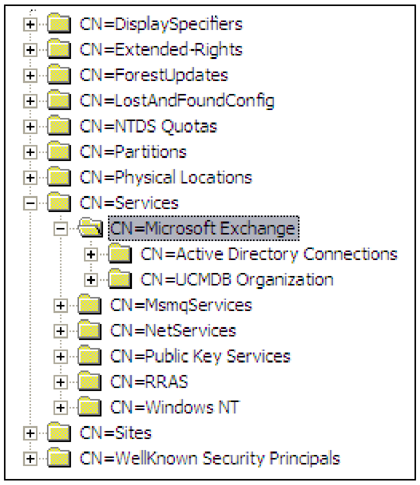
Note: To view the topology, see "Topology" on page 519.

Chapter 39: Microsoft Exchange Server with Active Directory Discovery

This chapter includes:

Overview

With the addition of LDAP protocol support in Content Pack 5, DFM can discover the Exchange topology using Active Directory (AD). Because Exchange is tightly integrated with AD and stores most of its configuration there, DFM connects to the AD Domain Controller and extracts information from it. The Exchange configuration is stored in a specific node under Services:



The Base Distinguished Name of this node is:

"CN=Microsoft Exchange, CN=Services, CN=Configuration,DC=ucmdb-ex, DC=dot"

where **ucmdb-ex.dot** is the name of the domain in this example.

If this node exists, DFM drills down and discovers all remaining information that includes: Exchange organization, Exchange servers, administrative and routing groups, connectors, roles, and so on.

Multiple Domain Controllers can serve the same domain, in which case the information is replicated between them (multi-master replication). The controllers contain the same data, so DFM needs to run only against one of them.

Note: The job for AD discovery triggers on, and runs against, all discovered domain controllers. However, as only updates are sent to the CMDB by the Data Flow Probe's result processing mechanism, the information is reported only once.

AD machines in the domain are registered in DNS as being configured for AD. DFM retrieves the FQDN (fully qualified domain name) from every Exchange discovery. This is the name of Exchange within AD. To report such an Exchange, DFM tries to resolve the FQDN to an IP address, as follows:

- DFM uses the default Data Flow Probe's DNS to resolve the Exchange FQDN.
- If this fails, DFM uses the target Domain Controller as the DNS. This is because in many cases the DNS server runs on the same machine as the Domain Controller. DFM runs the command **"nslookup <FQDN> <targetDC>"** in the Data Flow Probe's local Shell.
- If this fails, DFM skips this Exchange instance.

Note: If the FQDN cannot be resolved either by a local DNS or by using the target Domain Controller as the DNS, the job displays the following message:

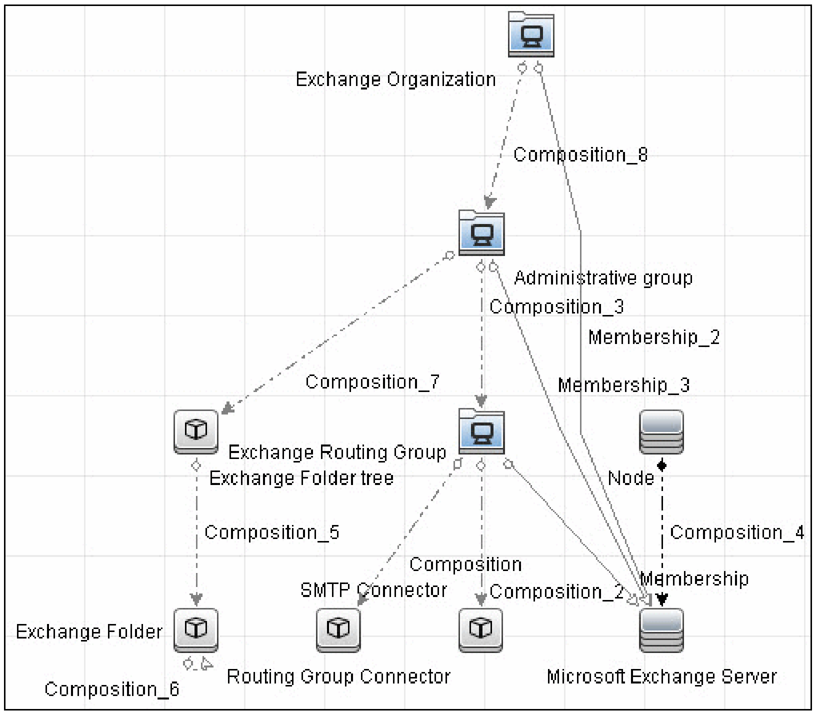
```
Cannot resolve IP address for host '<host>', Exchange Server won't be reported
```

Supported Versions

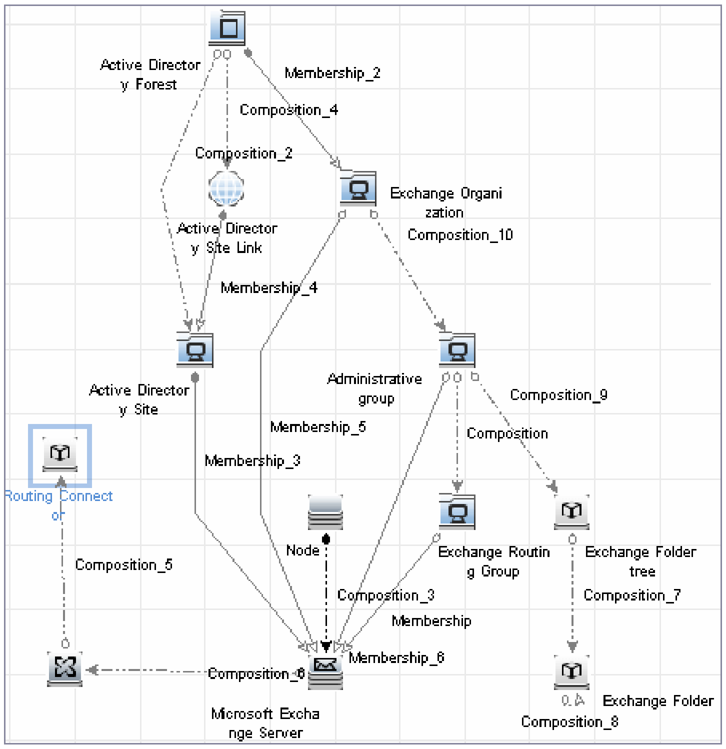
Microsoft Exchange discovery with Active DIrectory supports MS Exchange versions 2003, 2007, and 2010.

Topology

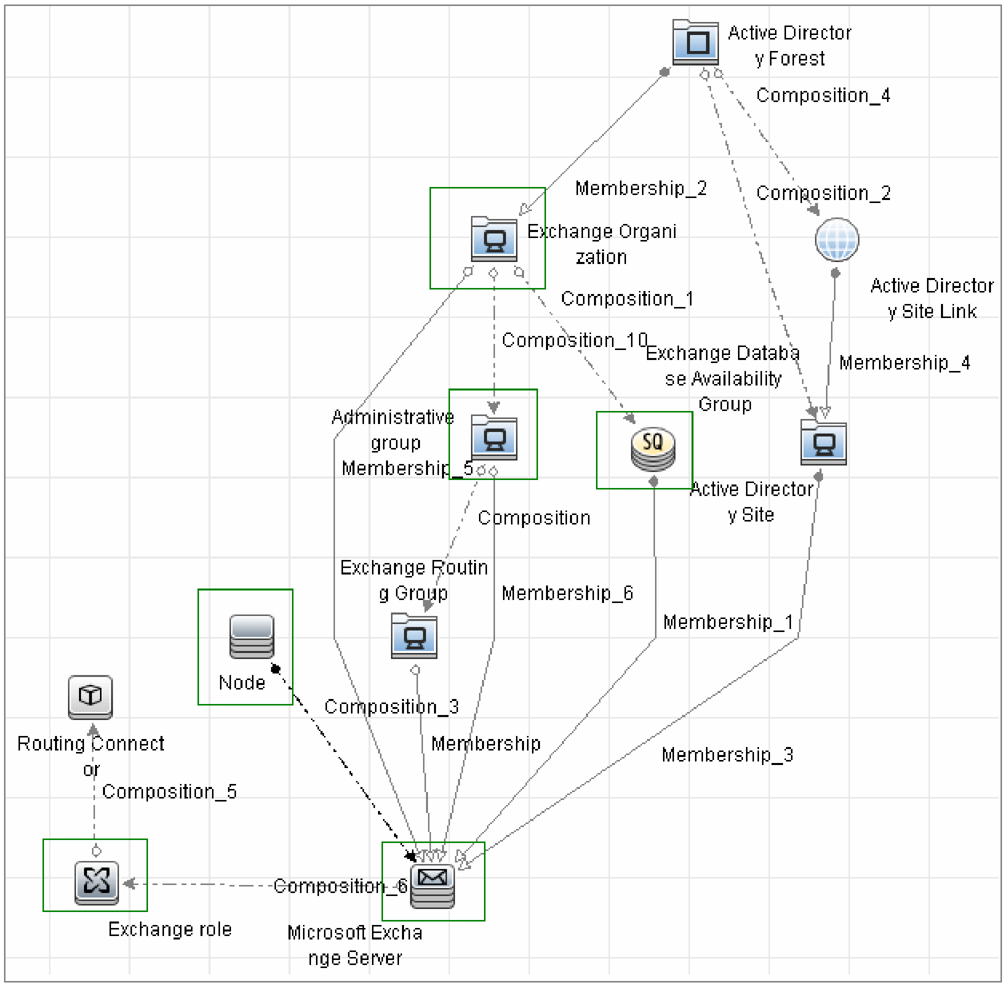
- **Microsoft Exchange Server 2003**



- **Microsoft Exchange Server 2007**



- **Microsoft Exchange Server 2010**



How to Discover Microsoft Exchange Server Topology with Active Directory

Note: This functionality is available as part of Content Pack 5.00 or later.

This section explains how DFM discovers Exchange by utilizing the tight integration between Exchange and AD. DFM runs jobs to discover Exchange elements in the topology that are available only through AD.

This task includes the following steps:

1. **Prerequisite – Set up protocol credentials**

   Define at least one set of LDAP protocol credentials. These credentials should enable connecting to a Domain Controller through the LDAP protocol and performing searches. DFM does not modify information in AD. The queried nodes reside in the Configuration partition under the following nodes:

   - **CN=Services,CN=Microsoft Exchange** node
   - **CN=Sites** node

   The LDAP protocol credentials should include:

   - **User name** and **password.** Use the user account from the target domain. For all nodes that are to be queried, give **List Contents** and **Read all properties** permissions.
   - **Authentication type**. **Simple**.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite – Discover a Domain Controller**

   To discover the Exchange topology with AD, DFM must first find a Domain Controller with an available LDAP connection.

   a. Activate the **Range IPs by ICMP** job, to ping the target host on which the Domain Controller runs.

   b. Activate the **TCP Ports** job against the target host, to discover open LDAP ports.

c. Activate the **Active Directory Connection by LDAP** job, to discover the Domain Controller on the target host.

d. To enable DFM to use the LDAP protocol, edit the following line in the **portNumberToPortName.xml** file (**Adapter Management > Resources pane > Packages > DDMInfra > Configuration Files**).

Change:

```
<portInfo portProtocol="tcp" portNumber="389" portName="ldap" discover="0"
/>
```

to

```
<portInfo portProtocol="tcp" portNumber="389" portName="ldap" discover="1"
/>
```

3. **Run the discovery**

Activate the **Microsoft Exchange Topology by LDAP** job.

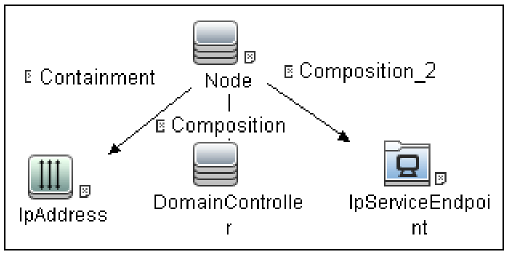
Microsoft Exchange Topology by LDAP Job

The components responsible for discovering Microsoft Exchange Server with Active Discovery are bundled in the Microsoft Exchange Server package, **Microsft_exchange_server.zip**.

Trigger Query

- Trigger CI: DomainController
- Trigger query:

  The Trigger query, **trigger_domainctl_ldap**, is part of the Active Directory package.



- CI attribute conditions:

CI	Attribute Value
IpAddress	NOT IP Probe Name Is null
DomainController	NOT Reference to the credentials entry dictionary Is null AND NOT Application IP Is null
IpServiceEndpoint	`Name Equal ignore case ldap`

Adapter

This discovery uses the **ms_exchange_topology_by_ldap** adapter.

- Created/Changed CITs

Additional CITs	The following CITs have been added to the Microsoft Exchange Server Package ▪ Routing Group Connector ▪ SMTP Connector ▪ Exchange Routing Connector ▪ Send Connector ▪ Receive Connector ▪ Exchange Storage Group ▪ Exchange Mailbox Database ▪ Exchange Routing group
Deprecated CITs	The following CITs are deprecated; they remain in the package but are no longer reported: ▪ Directory Service Access DC ▪ Exchange Message queue ▪ Exchange link
Modified CITs	The following CITs were modified: ▪ **Exchange System** is now **Exchange Organization** ▪ **Microsoft Exchange Server** includes a new attribute: **is_master**

Discovered CITs

- Active Directory Forest
- Active Directory Site
- Active Directory System

- Administrative Group
- Composition
- Containment
- Exchange Database Availability Group
- Exchange Folder
- Exchange Folder Tree
- Exchange Mailbox Database
- Exchange Organization
- Exchange Role
- Exchange Routing Connector
- Exchange Routing Group
- ExecutionEnvironment
- Host
- IpAddress
- Membership
- Microsoft Exchange Server
- Ownership
- Routing Group Connector
- SMTP Connector

Troubleshooting and Limitations

Currently Exchange Folders are not reported through the **Microsoft Exchange Topology by LDAP** job.

Chapter 40: Microsoft Exchange Server Discovery by NTCMD or UDA

This chapter includes:

Overview

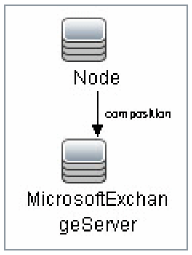
DFM discovers the following components of Microsoft Exchange Server (Exchange) software: Microsoft Exchange Server, Server Roles, Administrative and Routing groups, Organization, Clustered Mail Box, Database Availability group, Public folders, and Folder trees.

Supported Versions

Microsoft Exchange Server Discovery by NTCMD or UDA supports MS Exchange Server version 2007, 2010.

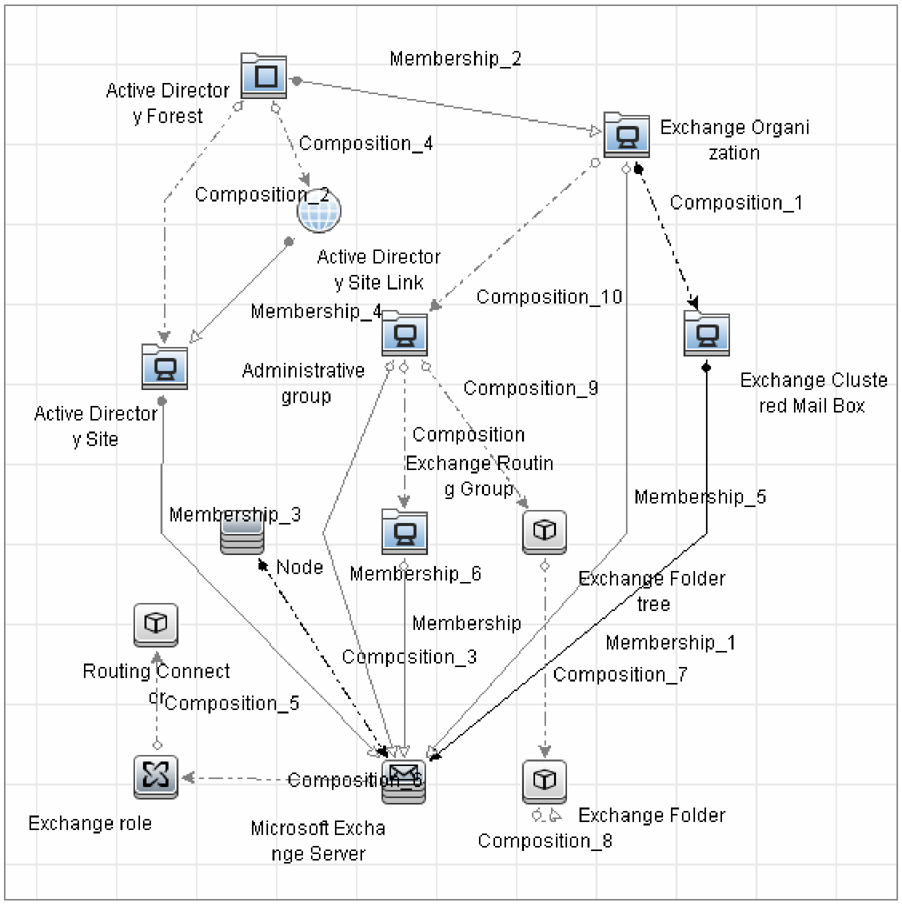
Topology

MS Exchange Connection by NTCMD or UDA:

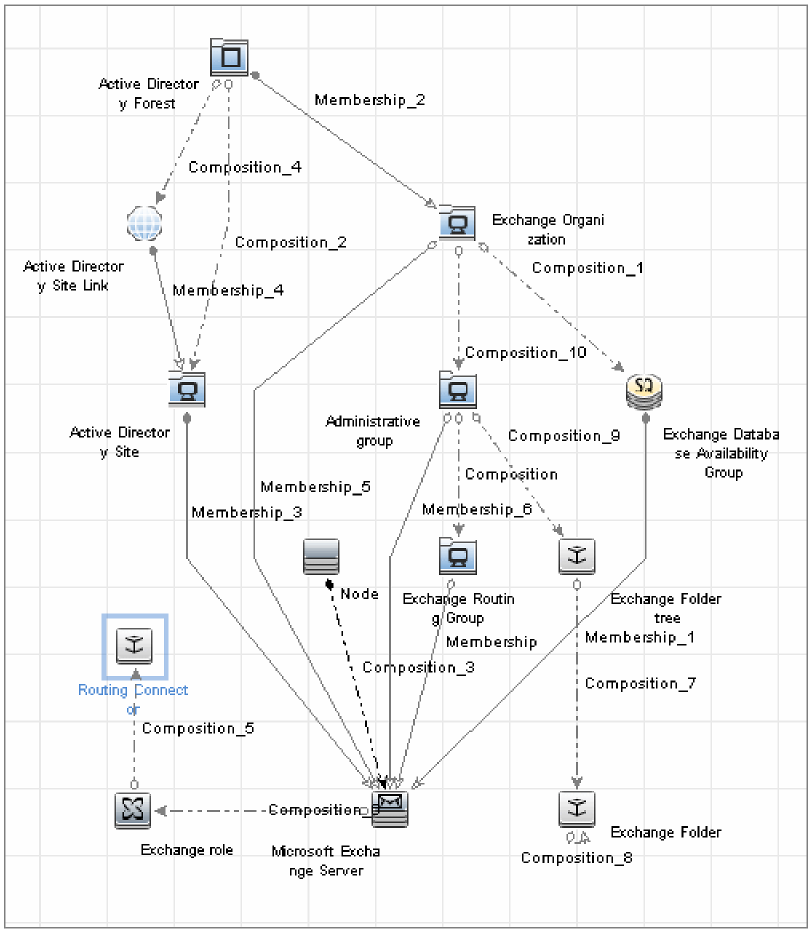


MS Exchange 2007 Topology:

DFM runs the NTCMD protocol to retrieve the topology for MS Exchange 2007.



MS Exchange 2010 Topology:



How to Discover Microsoft Exchange Server by NTCMD or UDA

DFM discovers Exchange by executing a PowerShell script on a remote machine with Exchange installed.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery is based on the following protocol:

   - NTCMD protocol, or Universal Discovery protocol if UD Agent is installed on the Exchange server.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Set up permissions**

   - Set the script execution policy either to **Unrestricted** or **Remote Signed**.
   - Verify that the account used for discovery has the permissions of the **Exchange View-Only Administrator** role.

3. **Run the discovery**

   a. Run the **Host Connection by Shell** job.

   b. Run the **Host Applications by Shell** job to discover the Exchange process.

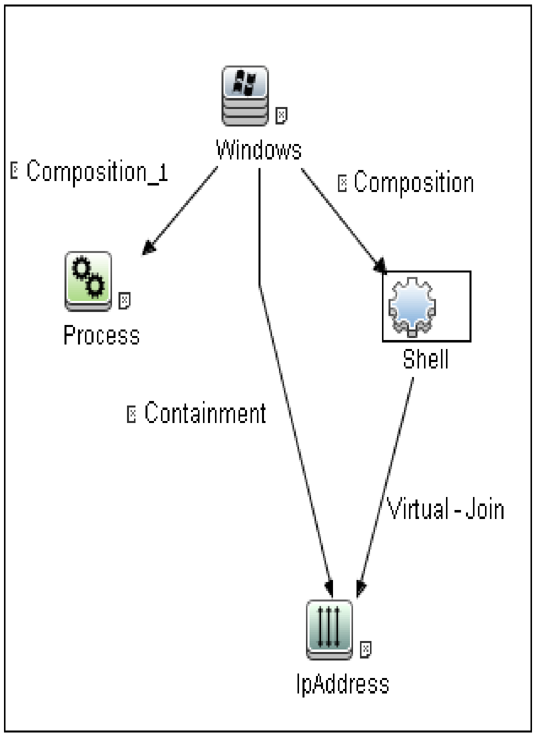
   c. Run the **Microsoft Exchange Connection by NTCMD or UDA** job to discover Exchange Server CIs.

   d. Run the **Microsoft Exchange Topology by NTCMD or UDA** job to discover the rest of the topology.

Microsoft Exchange Connection by NTCMD or UDA Job

This section contains details about the job.

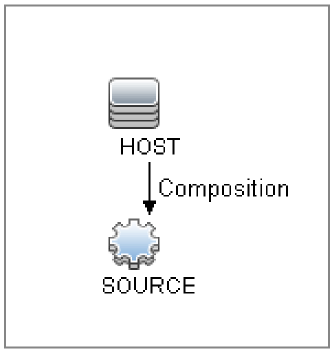
Trigger Query



Adapter

This job uses the **ms_exchange_connection_by_ntcmd** adapter.

- Input query:



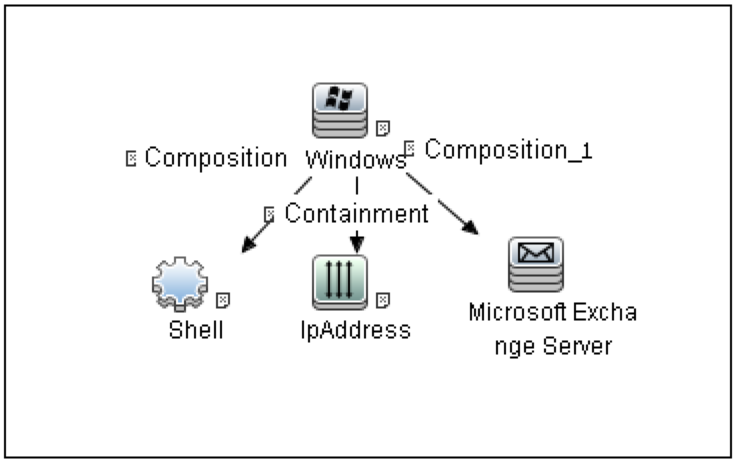
Discovered CITs

- Composition
- MicrosoftExchangeServer
- Node

Microsoft Exchange Topology by NTCMD or UDA Job

This section contains details about the job.

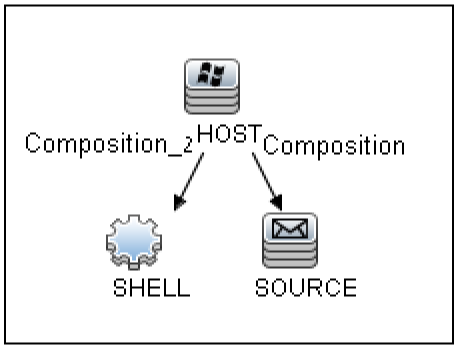
Trigger Query



Adapter

This job uses the **ms_exchange_topology_by_ntcmd** adapter.

- Input query:



Discovered CITs

- Administrative group
- Composition
- Exchange Client Access Server
- Exchange Clustered Mail Box

- Exchange Database Availability Group
- Exchange Edge Server
- Exchange Hub Server
- Exchange Mail Server
- Exchange Organization
- Exchange Unified Messaging Server
- Membership
- MicrosoftExchangeServer
- Node

Created/Changed CITs

The following CITs are used to create CIs for Exchange components:

Exchange Organization	This CIT represents the top-level Exchange system. For example, if an organization uses the Exchange solution, all the Exchange components are linked to a single Exchange Organization CI.
Microsoft Exchange Server	This CIT is inherited from the RunningSoftware CIT. The CIT represents Exchange software installed on a host.
Exchange Folder	This CIT represents Public folders available on the Exchange system. Public folder can be organized in a hierarchical structure, that is, one Public folder can contain another Public folder.

Exchange Role	This CIT is located in the **Application Resource > Microsoft Exchange Resource** folder. It is an abstract CIT that is the parent of the following CITs: • **Exchange Client Access Server**. Represents the Client Access Server role. • **Exchange Mail Server**. Represents the Mail Server role. • **Exchange Edge Server**. Represents Edge Server role. • **Exchange Hub Server**. Represents Hub Server role. • **Exchange Unified Messaging server**. Represents Unified Messaging server role.

Chapter 41: Microsoft Exchange Server by PowerShell Discovery

This chapter includes:

Overview

Microsoft Exchange Server is the server side of a client–server, collaborative application product developed by Microsoft. It is part of the Microsoft Servers line of server products and is used by enterprises using Microsoft infrastructure products. Exchange's major features consist of electronic mail, calendaring, contacts and tasks; support for mobile and web-based access to information; and support for data storage.

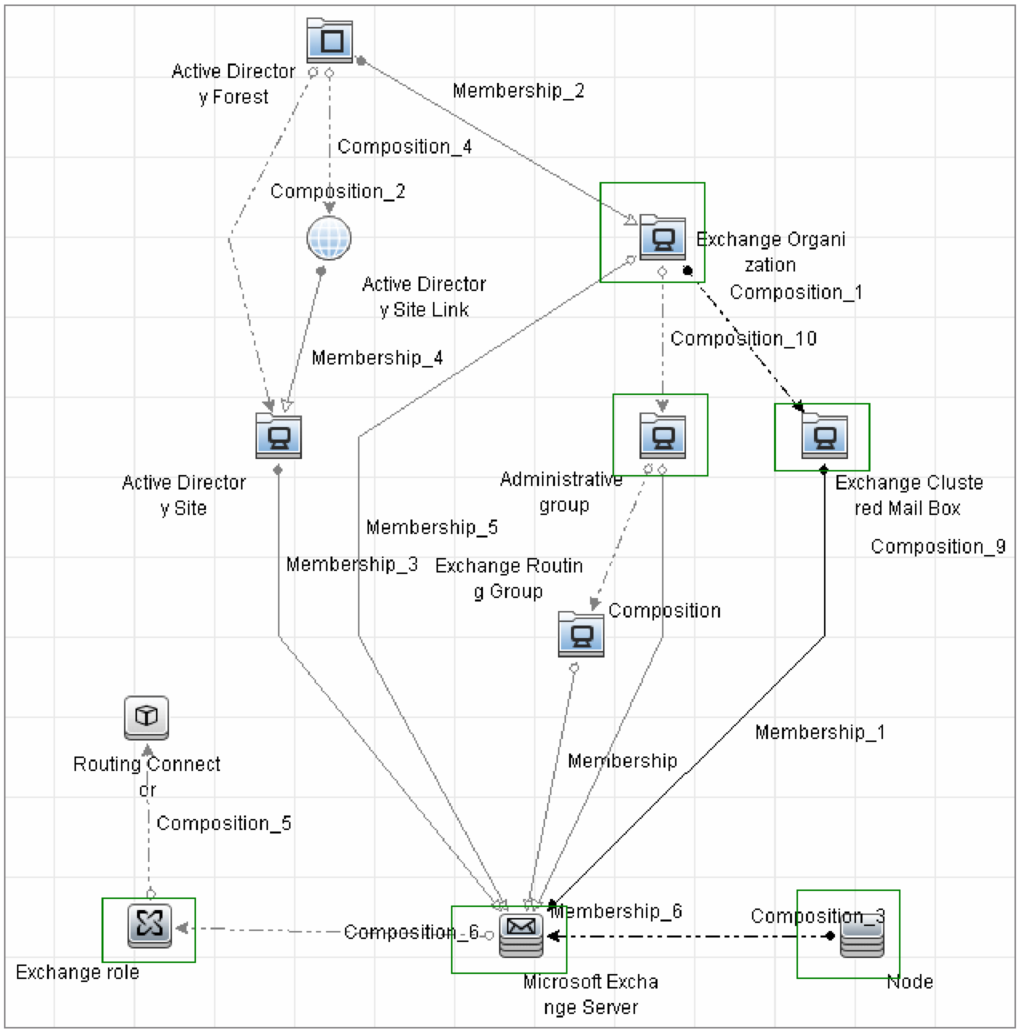
Supported Versions

Microsoft Exchange by PowerShell discovery supports MS Exchange Server versions 2007 and 2010.

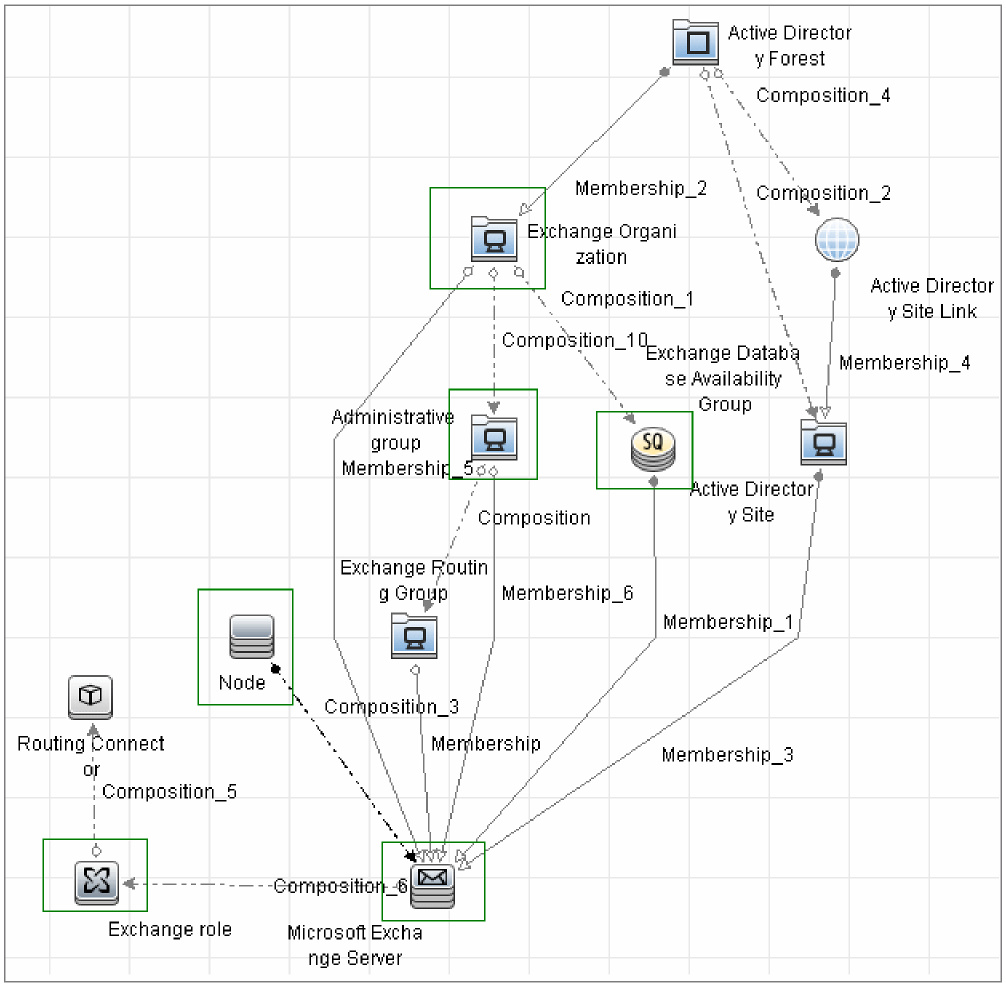
Topology

The following images illustrate the Microsoft Exchange by PowerShell topology. The CITs marked with borders can be discovered by the **Microsoft Exchange Topology by PowerShell** job.

- **Microsoft Exchange Server 2007 by PowerShell**



- **Microsoft Exchange Server 2010 by PowerShell**



How to Discover Microsoft Exchange by PowerShell

The following steps describe how to discover Microsoft Exchange by PowerShell.

1. **Prerequisite - Set up protocol credentials**

   This discovery solution is based on the PowerShell protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Before starting the discovery ensure that PowerShell v2.0 is installed on the Data Flow Probe machine.

2. **Prerequisite - Configure PowerShell remoting and AD**

   a. Enable PowerShell remote access. For details, see "How to Configure PowerShell Remoting" on the next page.

   b. Configure the Active Directory side. For details, see "How to Configure the Active Directory Side" on page 555.

3. **Prerequisite - Set up permissions**

   Before starting the discovery, ensure that the discovery user has been granted all the required permissions to run the following commands:

   - **Snap-Ins:**
     - Microsoft.Exchange.Management.PowerShell.Admin (Exchange 2007)
     - Microsoft.Exchange.Management.PowerShell.E2010 (Exchange 2010)
   - **Get-ClusteredMailboxServerStatus**
   - **Get-ExchangeServer**
   - **Get-DatabaseAvailablityGroup**
   - **hostname**

4. **Run the discovery**

a. Run the **Range IPs by ICMP** job to discover the Windows system IP addresses.

b. Run the **Host Connection by PowerShell** job to discover the Windows connection with the PowerShell agent and networking topology.

c. Run the **Host Applications by PowerShell** job to discover the host applications.

d. Run the **Microsoft Exchange Topology by PowerShell** job.

How to Configure PowerShell Remoting

This task describes how to enable PowerShell remote access.

This task includes the following steps:

1. **Launch the PowerShell configuration**

   In the PowerShell command prompt run the **winrm quickconfig**.

   > **Note:** From the moment that the PowerShell configuration is launched, you must differ between the server side configuration and client side configuration.

2. **Configure the server-side machine**

   On the server, depending on the authentication method that will be used, perform the following steps:

   a. Run **cd WSMan:\localhost\Service\Auth**

   b. Run **dir** and verify that the required authentication type is enabled, that is, the **State** = **True**. If the required authentication type is disabled, run "**et-Item <AuthTypeName> True**. By default, **Kerberos** and **Negotiate** are enabled.

   c. Run **cd WSMan:\localhost\Service** and verify that **IPv4Filter** or **IPv6Filter** are set to either "*" or to any other valid value for your environment.

   d. Run **cd WSMan:\localhost\Listener**, and then **dir**. Verify that the listener actually listens to the required IPs. By default, the listener listens to all IPs if the value "*" is used.

   e. If you made any changes, restart the **winrm service** by running the **restart-service winrm** command

3. **Configure the client-side machine**

   On the client machine, perform the following steps:

   a. Run **cd WSMan:\localhost\Client\Auth**

   b. Run **dir** and verify that the required authentication type is enabled, that is, the **State** = **True**. If the required authentication type is disabled, run **Set-Item <AuthTypeName> True**.

      Note: The allowed protocols must coincide with the ones configured on the server side.

   c. Run **cd WSMan:\localhost\Client**.

   d. Run **dir** and check value of **TrustedHosts**. By default, the value is empty so that no connection outside is possible. **TrustedHosts** is an ACL field where the allowed values are a domain name or a list of domain names and an IP address or a list of IP addresses. The value may have a special symbol **""**, meaning that any destination or any symbol can appear in any part of the specified destinations list. If the only value is **""**, then the client is allowed to connect to any host. This is the recommended value.

      To change the value for **TrustedHosts**, use **Set-Item TrustedHosts <Value>**.

      Note: No translation from FQDN to IP is done while validating the ACL. This means that if the connection is performed by IP and only an FQDN is listed in the **TrustedHosts** field (or vice versa), the connection will not be allowed.

   e. If you made any changes, restart the **winrm service** by running the **restart-service winrm** command.

How to Configure the Active Directory Side

Some Exchange PowerShell command-lets need to perform AD LookUps. AD servers (starting from Win 2003) do not allow **Anonymous** lookups while the impersonalization is still applied. This results in various errors while trying to run the Exchange/AD-related command-lets remotely.

This task includes the following steps:

1. **Configure delegation on the Active Directory side**

   To enable remote calls of such command-lets, you must configure the **Delegation** on the Active Directory side.

   a. Log onto the domain controller using an administrator account.

   b. Select **Start > Programs > Administrative Tools > Active Directory Users and Computers**.

   c. Select you domain's, **Users** folder.

   d. Right-click the user account that is to be delegated, and click **Properties**.

   e. In the **Account** tab, under the **Account options**, make sure that the **Account is sensitive and cannot be delegated** option is NOT selected.

   f. Click **OK**.

2. **Allow required servers to perform the delegated requests**

   Confirm that the server process account is trusted for delegation if the server process runs under a Windows user account:

   a. In the **Active Directory Users and Computers > Users** folder, right-click the user account that is used to run the server process that will impersonate the client, and click **Properties**.

   b. In the Account tab, under the **Account options**, select the **Account is trusted for delegation** option.

3. **Confirm that the server process account is trusted for delegation for the server process**

   a. In **Active Directory Users and Computers,** right-click **Computers**, and click **Properties**.

   b. Right-click the server computer (where the process that impersonates the client will be running), and click **Properties**.

   c. On the General page, select **Trust computer for delegation**.

   d. Select **Use any authentication protocol**.

   e. Click **Add** and select the required processes.

   f. If only the Kerberos protocol is used, select the **Trust this computer for delegation to any**

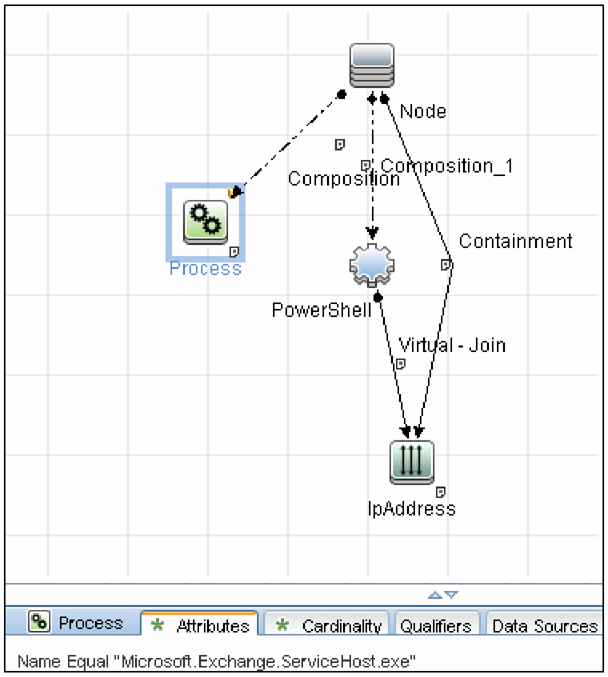
service or **Use Kerberos only**.

Note: If the **Kerberos** authentication is used and the connection is performed from outside of the destination domain, **Trust Domain** must be configured on the target AD.

Microsoft Exchange Topology by PowerShell Job

The components responsible for discovering Microsoft Exchange Server by PowerShell are bundled in the Microsoft Exchange Server package, **Microsft_exchange_server.zip**.

Trigger Query

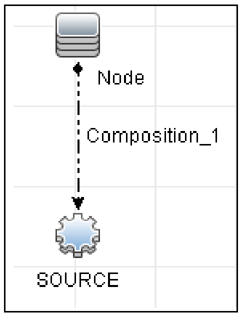


Name Equal "Microsoft.Exchange.ServiceHost.exe"

Adapter

This job uses the **MS_Exchange_Topology_by_Powershell** adapter.

- Input Query



- Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

- Used Scripts

  The following scripts are used by Microsoft Exchange by PowerShell discovery.

  - host_win.py
  - host_win_shell.py
  - ms_exchange.py
  - ms_exchange_topology_by_powershell.py
  - ms_exchange_win_shell.py
  - networking_win_shell.py

Created/Changed Entities

Entity Name	Entity Type	Entity Description
Microsoft Exchange Topology by PowerShell.xml	Job	Main Job
MS_Exchange_Topology_by_PowerShell.xml	Adapter	Discovery adapter
ms_exchange_topology_by_powershell.py	Script	Discovery script
ms_exchange_process_and_powershell.xml	TQL	Trigger Query
ms_exchange_clustered_mailbox.xml	Class	CI Type
ms_exchange_dag.xml	Class	CI Type
ms_exchange_win_shell.py	Script	Discovery script
ms_exchange.py	Script	Discovery script

Commands

The following commands are used by Microsoft Exchange by PowerShell discovery.

- **Get-ExchangeServer Command**

  ```
  Get-ExchangeServer | Where-Object {$_.Fqdn.ToLower().StartsWith
  ((hostname).ToLower())} | Format-List Name, Guid, Fqdn, ServerRole, DataPath,
  WhenCreated, ExchangeVersion, AdminDisplayVersion, OrganizationalUnit, Site,
  ExchangeLegacyDN
  ```

  - Output

    ```
    Name : SAM-RND-DC01
    Guid : e8f5c340-6cf1-4fc6-aa34-226ab99282dd
    Fqdn : SAM-RND-DC01.ddm-rnd.ua
    ServerRole : Mailbox, ClientAccess, UnifiedMessaging, HubTransport
    DataPath : C:\Program Files\Microsoft\Exchange Server\V14\Mailbox
    WhenCreated : 8/6/2010 5:24:05 PM
    ExchangeVersion : 0.1 (8.0.535.0)
    AdminDisplayVersion : Version 14.0 (Build 639.21)
    OrganizationalUnit : ddm-rnd.ua/SAM-RND-DC01
    Site : ddm-rnd.ua/Configuration/Sites/Default-First-Site-Name
    ExchangeLegacyDN : /o=SiteScope Rnd Lab/ou=Exchange Administrative Group
    (FYDIBOHF23SPDLT)/cn=Configuration/cn=Servers/cn=SAM-RND-DC01
    ```

  - Mapping

    The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
Name	Exchange Server	Name
Guid	Exchange Server	Guid
Fqdn	Exchange Server	Fqdn
ServerRole	Corresponding Server Role CIs are created	Corresponding Server Role CIs are created
WhenCreated	Exchange Server	Creation Date
ExchangeLegacyDN	Exchange Server	Organization

Command Output Attribute	CI Type	CI Attribute
AdminDisplayVersion	Exchange Server	Version
AdminDisplayVersion	Exchange Server	Application Version
AdminDisplayVersion	Exchange Server	Application Version Description

- **Get-ClusteredMailboxServerStatus Command**

  ```
  Get-ClusteredMailboxServerStatus
  ```

  - Output

    ```
    Identity : ddm-ex2k7ccr
    ClusteredMailboxServerName : DDM-EX2K7CCR.ddm01.local
    State : Online
    OperationalMachines : {DDM-EX2K7CCR-N1 <Active, Quorum Owner>,
    DDM-EX2K7CCR-N2}
    FailedResources : {}
    OperationalReplicationHostNames : {ddm-ex2k7ccr-n1, ddm-ex2k7ccr-n2}
    FailedReplicationHostNames : {}
    InUseReplicationHostNames : {ddm-ex2k7ccr-n1, ddm-ex2k7ccr-n2}
    IsValid : True
    ObjectState : Unchanged
    ```

  - Mapping

    The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
Identity	Exchange Clustered Mailbox	Name
ClusteredMailboxServerName	Used to determine the name of the cluster	Used to determine the name of the cluster

- **Get-DatabaseAvailabilityGroupCommand**

  ```
  Get-DatabaseAvailabilityGroup | format-list
  ```

  - Output

    ```
    Name : DDMDAG
    Servers : {DDM-EXCLN2, DDM-EXCLN1}
    ```

```
WitnessServer : DDM-EXCLDC.DDM.LOCAL
WitnessDirectory : c:\EXCLFSW
AlternateWitnessDirectory :
NetworkCompression : InterSubnetOnly
NetworkEncryption : InterSubnetOnly
DatacenterActivationMode : Off
StoppedMailboxServers : {}
StartedMailboxServers : {}
DatabaseAvailabilityGroupIpv4Addresses : {172.24.10.129}
OperationalServers :
PrimaryActiveManager :
ThirdPartyReplication : Disabled
ReplicationPort : 0
NetworkNames : {}
AdminDisplayName :
ExchangeVersion : 0.10 (14.0.100.0)
DistinguishedName : CN=DDMDAG,CN=Database Availability
Groups,CN=Exchange Administrative Group
(FYDIBOHF23SPDLT),CN=Administrative Groups,CN=Discovery,CN=Microsoft
Exchange,CN=Services,CN=Configuration,DC=ddm, DC=local
Identity : DDMDAG
Guid : 51799b4d-9c0d-4842-990a-f9862be3e7a4
ObjectCategory : ddm.local/Configuration/Schema/ms-Exch-MDBAvailability-
Group
ObjectClass : {top, msExchMDBAvailabilityGroup}
WhenChanged : 1/31/2011 4:24:34 PM
WhenCreated : 1/31/2011 3:45:06 PM
WhenChangedUTC : 1/31/2011 2:24:34 PM
WhenCreatedUTC : 1/31/2011 1:45:06 PM
OrganizationId :
OriginatingServer : ddm-excldc.ddm.local
IsValid : True
```

- Mapping

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	**CI Type**	**CI Attribute**
Name	Exchange Database Availability Group	Name
Distinguished name	Used to relate to an Exchange organization	Used to relate to an Exchange organization

Discovered CITs

- Composition
- Exchange Client Access Server
- Exchange Clustered Mail Box
- Exchange Database Availability Group
- Exchange Edge Server
- Exchange Hub Server
- Exchange Mailbox Database
- Exchange Mail Server
- Exchange Organization
- Exchange Unified Messaging Server
- Membership
- MicrosoftExchangeServer
- Node

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Microsoft Exchange Server by PowerShell discovery.

- **Problem:** No results brought, cmdlet calls end with errors like:

```
Active Directory error 0x80072020 occurred while searching for domain controllers in domain <Domain Name>: An operations error occurred.

+CategoryInfo :

+FullyQualifiedErrorId : 7D2B0C9D
```

Reason: The "Delegation" is not configured properly.

Solution: Configure Active Directory "Delegation" as described in "How to Configure the Active Directory Side" on page 555.

- **Problem:** No results brought, cmdlet calls end with errors like:

```
Value cannot be null..

Parameter name: parameters

 + CategoryInfo :

 + FullyQualifiedErrorId : System.ArgumentNullException,Microsoft.Exchange.Management.
SystemConfigurationTasks.GetExchangeServer
```

  Reason: The "Delegation" is not configured properly or connection is performed from an untrusted domain or not all required patches are installed on the server (for more details, see the official Microsoft site).

  Solution: Configure Active Directory "Delegation" as described in "How to Configure the Active Directory Side" on page 555, and check the patch-level. For more information, check the official Microsoft site.

- **Problem:** Calls to the Exchange command-lets fail with timeouts and/or session gets broken.

```
An application cannot impersonate a user and then run Windows PowerShell commands in
an Exchange Server 2007 environment.
```

  Reason: This is a known Exchange 2007 bug.

  Solution: To fix this problem, run Microsoft Patch KB943937, which is a part of MS Exchange 2007 SP1. For more information, see the Microsoft Patch description (http://support.microsoft.com/kb/943937).

Chapter 42: Microsoft Exchange Server by WMI Discovery

This chapter includes:

Overview

DFM discovers the following components of Microsoft Exchange Server (Exchange) software, versions 2003: Microsoft Exchange Server, Administrative and Routing groups, Organization, Public folders, and Folder trees.

All information about Exchange is retrieved by the WMI protocol from the **root\MicrosoftExchangeV2** namespace.

There are two jobs responsible for Exchange discovery:

- Microsoft Exchange Connection by WMI
- Microsoft Exchange Topology by WMI

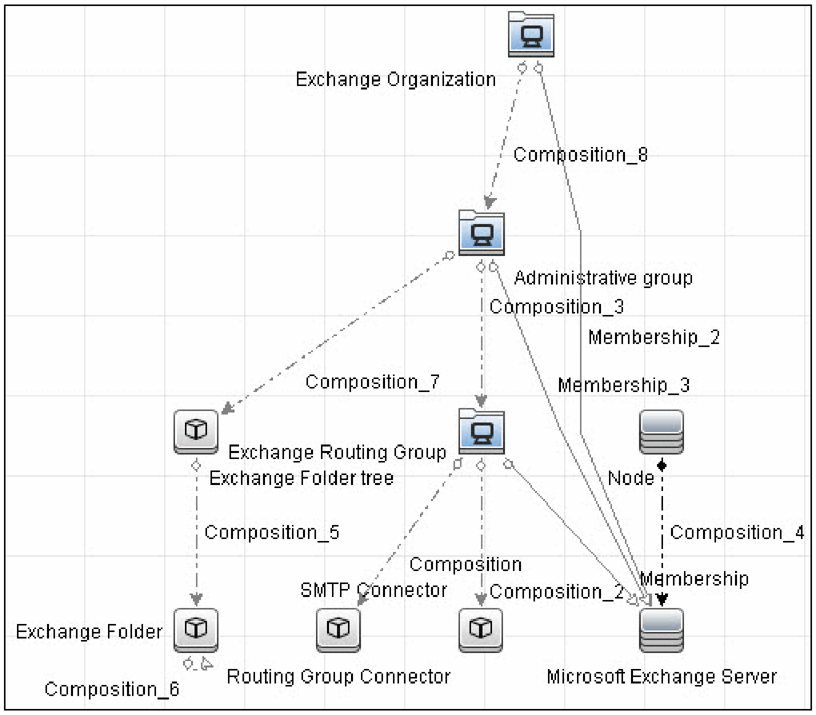
Supported Versions

Microsoft Exchange Server 2003

Topology

Microsoft Exchange Topology by WMI job

DFM connects to the remote host and retrieves the topology for MS Exchange 2003:



How to Discover Microsoft Exchange Server 2003 by WMI

This task explains how to discover MS Exchange Server 2003 using the WMI protocol.

1. **Prerequisite - Set up protocol credentials**

   This discovery is based on the WMI protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Information about Exchange is taken from the **root\MicrosoftExchangeV2** namespace.

2. **Prerequisite - Set up permissions**

   You must enable read-only permissions for the **root\MicrosoftExchangeV2 WMI** namespace. In some cases the **root\cimv2** namespace is also needed (with read-only permissions). For details, see "Troubleshooting and Limitations" on page 573.

3. **Run the discovery**

   Activate the following jobs:

   - **Network Discovery:**

     i. Run **Basic** > **Host Connection by WMI** to discover WMI CITs.

     ii. Run any of the **Host Resources and Applications** jobs that gather information about processes running on a host. If a process named **emsmta.exe** or **exmgmt.exe** is discovered on a host, the **Microsoft Exchange Connection by WMI** job is triggered.

   - **Enterprise Application > Microsoft Exchange**

     i. Run **Microsoft Exchange Connection by WMI**. This job reports the server that is actually running on this host. To discover other Exchange servers, you must run this job on each host where Exchange is running. The job creates Exchange CITs.

        This job connects to the remote host by WMI to the **root\MicrosoftExchangeV2** namespace.

        The following WMI queries are executed:

```
SELECT AdministrativeNote, CreationTime, ExchangeVersion, FQDN, GUID,
MTADataPath, MessageTrackingEnabled, MessageTrackingLogFileLifetime,
MessageTrackingLogFilePath, MonitoringEnabled, Type FROM Exchange_Server
```

This query returns all Exchange servers present in the Exchange organization.

ii. The Exchange CI created by **Microsoft Exchange Connection by WMI** job acts as a trigger for the **Microsoft Exchange Topology by WMI** job. The Trigger CI connects to the host where Exchange is running and retrieves the complete topology. (For details on troubleshooting error messages, see "Troubleshooting and Limitations" on page 573.)

This job connects to the remote host by WMI to the **root\MicrosoftExchangeV2** namespace. The following WMI queries are executed (order is preserved):

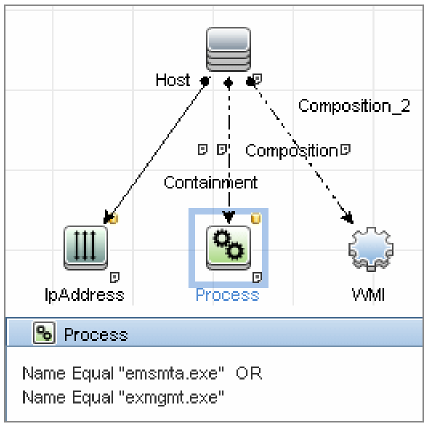
```
SELECT AdministrativeGroup, DN, FQDN, Name, RoutingGroup FROM Exchange_
Server
SELECT AdministrativeGroup, AdministrativeNote, CreationTime,
Description, GUID, Name, RootFolderURL FROM Exchange_FolderTree
SELECT AddressBookName, AdministrativeNote, Comment, ContactCount,
FolderTree, FriendlyUrl, IsMailEnabled, Path, Url FROM Exchange_
PublicFolder
```

Microsoft Exchange Connection by WMI Job

This section contains details about the job.

Trigger Query

- Trigger CI: ms_exchange_process_and_wmi
- Trigger query:

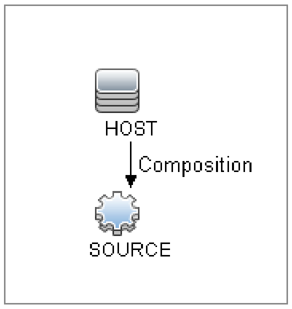


Name Equal "emsmta.exe" OR
Name Equal "exmgmt.exe"

Adapter

This job uses the **MS_Exchange_Connection_by_WMI** adapter.

- Input query:



Discovered CITs

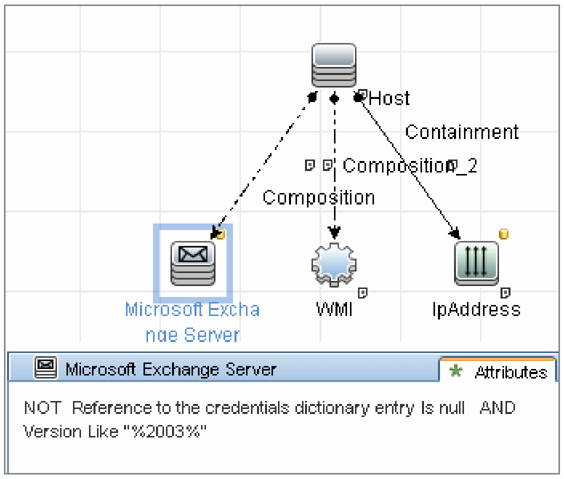
- Composition
- Computer
- MicrosoftExchangeServer

Microsoft Exchange Topology by WMI Job

This section contains details about the job.

Trigger Query

- Trigger CI: **ms_exchange_server_and_host_and_wmi**
- View: Microsoft Exchange Topology
- Trigger query:

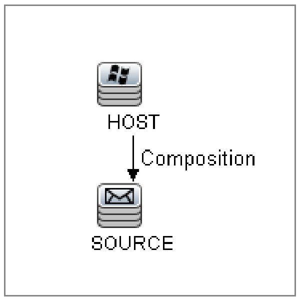


NOT Reference to the credentials dictionary entry Is null AND Version Like "%2003%"

Adapter

This job uses the **MS_Exchange_Topology_by_WMI** adapter.

- Input query:



Discovered CITs

- Administrative Group
- Composition
- Containment
- Exchange Folder
- Exchange Folder tree
- Exchange Organization
- Exchange Routing Group
- IpAddress
- Membership
- Node

Created/Changed CITs

The following CITs are created for Exchange components:

CIT	Description
Exchange	This CIT is located in the Application System folder. It is an abstract CIT that is the parent of the following CITs: • **Administrative group**. This CIT represents the administrative group in the Exchange organization. • **Exchange Organization**. This CIT represents the top-level of the Exchange organization. For example, if an organization uses the Exchange solution, then all the Exchange components are linked to a single Exchange Organization CI. • **Exchange Routing Group.** This CIT represents a Routing Group that exists in the Exchange organization. Routing groups supply varying network connectivity across servers, and restrict access of users in specific areas. Routing groups are deprecated in Exchange 2007. Instead Exchange 2007 relies on the Active Directory Sites configuration to connect between different Exchange Servers.
Microsoft Exchange Server	This CIT is inherited from the RunningSoftware CIT. The CIT represents Exchange software installed on a host.
Microsoft Exchange Resource	This CIT is located in the Application Resource folder. It is an abstract CIT that is the parent of the following CITs: • **Exchange folder.** This CIT represents the public folders available in the Exchange organization. A public folder may be organized in an hierarchical structure, that is, one public folder may contain another public folder. • **Exchange folder tree.** This CIT provides information about public and private folder trees on Exchange servers.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Microsoft Exchange by WMI discovery.

- **Administrative Group Limitation**.

  If an Administrative group does not contain any Exchange servers or folder trees, the Administrative group is not discovered.

- **Error Messages**

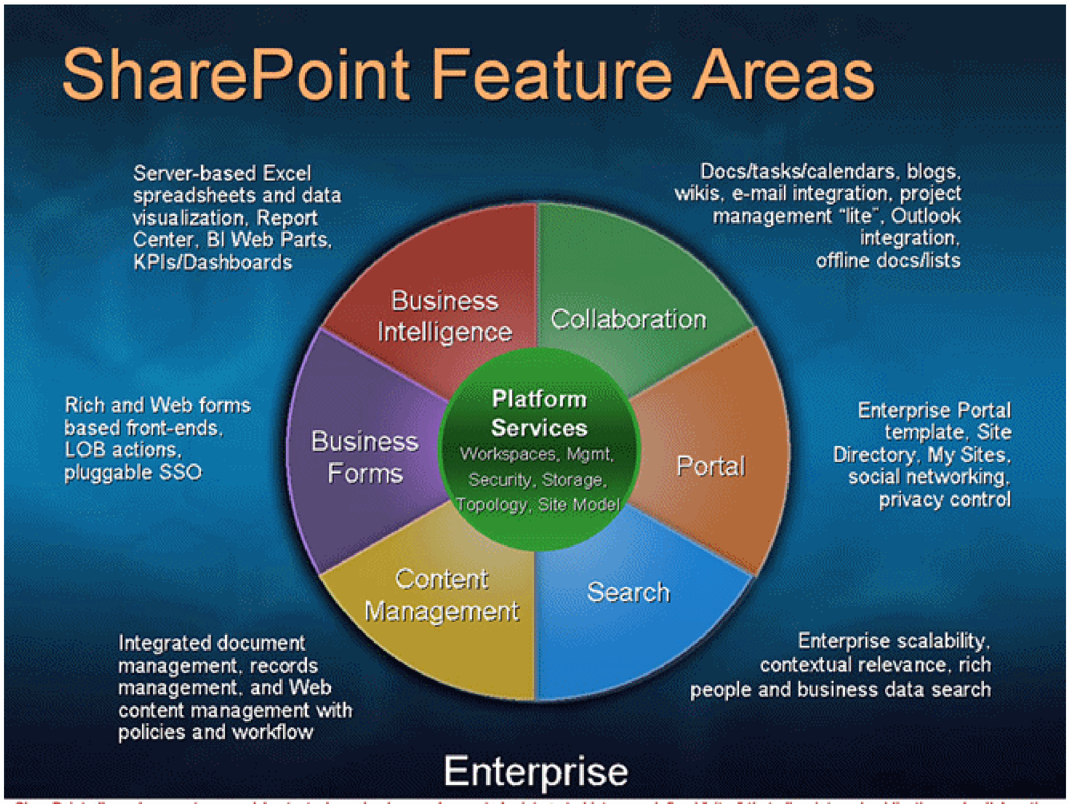
Error message	**Reason**	**Solution**
Failed to obtain host name	To model Exchange topology correctly, the **Microsoft Exchange Connection by WMI** job should know the name of the host to which it is connected. DFM tries to retrieve the **host_hostname** attribute of the host, matched by the input query. If the attribute is not set, DFM runs the following WMI query to obtain the domain name of the host: `SELECT Name FROM Win32_ComputerSystem` If this query fails for any reason, the job also fails with this error message.	■ Run any job that will retrieve the correct host name. ■ Set the host name manually. ■ Refer to the log files for more information as to why the WMI query for host name failed.
Failed to discover folder trees and public folders		Check if the credentials you use for connection match those described in "Prerequisite - Set up protocol credentials" on page 568.

Chapter 43: Microsoft SharePoint Discovery

This chapter includes:

Overview

Microsoft SharePoint is a family of software products developed by Microsoft for collaboration, file sharing, and Web publishing. This family of products include: Microsoft SharePoint Server, Microsoft SharePoint Foundation, Microsoft Search Server, Microsoft SharePoint Designer, and Microsoft SharePoint Workspace.


SharePoint Feature Areas
Server-based Excel spreadsheets and data visualization, Report Center, BI Web Parts, KPIs/Dashboards
Docs/tasks/calendars, blogs, wikis, e-mail integration, project management "lite", Outlook integration, offline docs/lists
Business Intelligence
Collaboration
Rich and Web forms based front-ends, LOB actions, pluggable SSO
Business Forms
Platform Services
Workspaces, Mgmt, Security, Storage, Topology, Site Model
Portal
Enterprise Portal template, Site Directory, My Sites, social networking, privacy control
Content Management
Search
Integrated document management, records management, and Web content management with policies and workflow
Enterprise scalability, contextual relevance, rich people and business data search
Enterprise

SharePoint allows documents, spreadsheets, tasks, calendars, and more to be integrated into user-defined "sites" that allow internal publication and collaboration between team members. Add document management, workflow, digital rights management, and Outlook integration, and you have an incredibly rich environment for sharing information within and organization.

In terms of the CMDB class model, it can be described as a set of services (application server, search server, indexing server, and so on) with its Web tier based on IIS, and its storage tier based on the MS SQL Server.

Supported Versions

Microsoft SharePoint discovery supports:

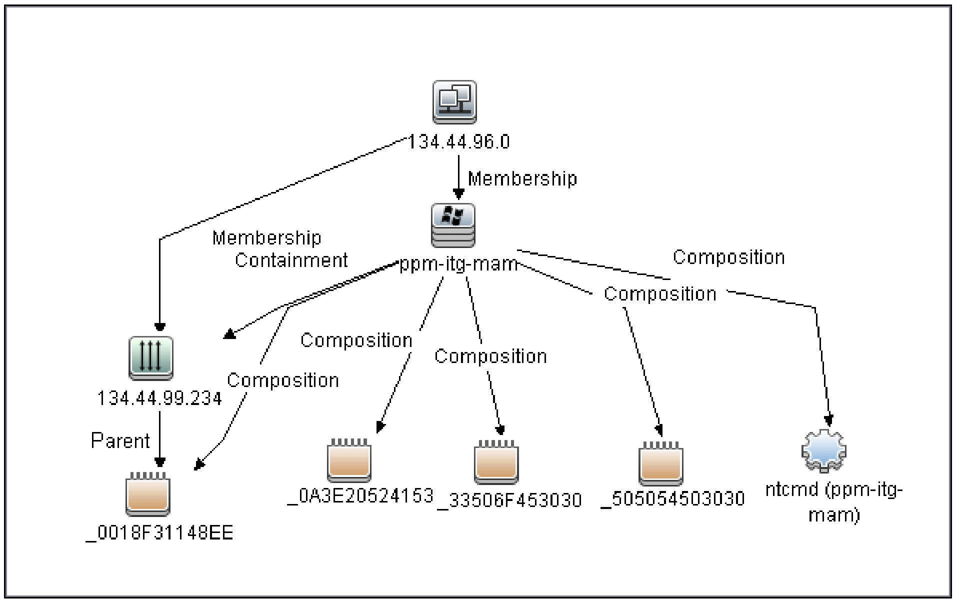
- Microsoft SharePoint 2007
- Microsoft SharePoint Server 2010

Note: This discovery is expected to work on all available versions of Microsoft SharePoint.

Topology

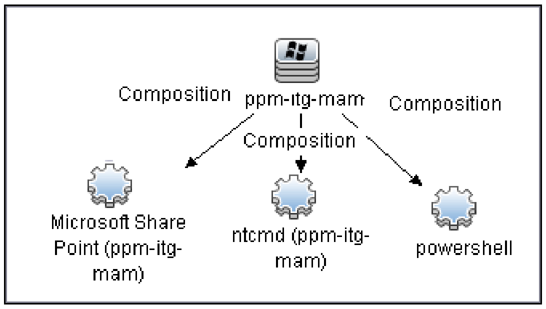
The following images display sample output for the Sharepoint discovery jobs.

Host Connection by Shell Job



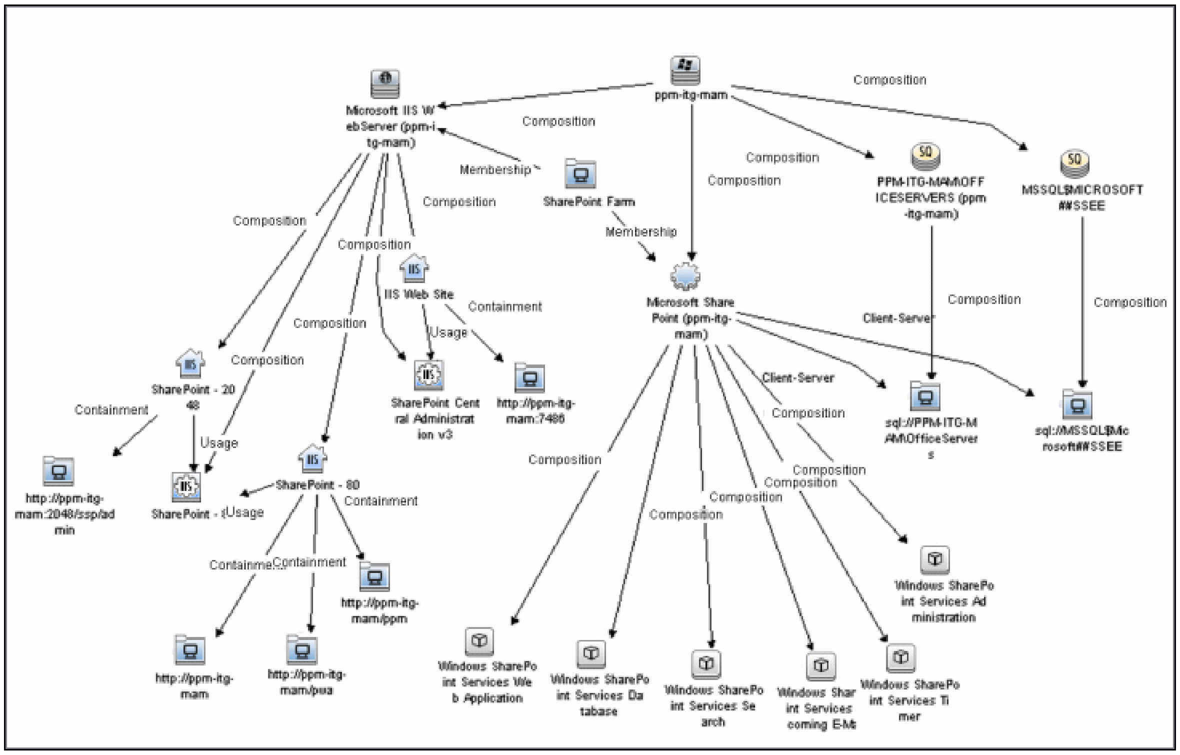
Host Applications by Shell Job

Note: Only the data necessary for the continued flow is shown.


ppm-itg-mam
Composition
Composition
Composition
Microsoft Share Point (ppm-itg-mam)
ntcmd (ppm-itg-mam)
powershell

Microsoft SharePoint Topology Job

Note: For a list of discovered CITs, see "Discovered CITs" on page 583.


ppm-itg-mam
Microsoft IIS WebServer (ppm-itg-mam)
SharePoint Farm
PPM-ITG-MAM\OFFICESERVERS (ppm-itg-mam)
MSSQL$MICROSOFT##SSEE
IIS Web Site
Microsoft SharePoint (ppm-itg-mam)
SharePoint - 2048
SharePoint Central Administration v3
http://ppm-itg-mam:7486
sql://PPM-ITG-MAM\OfficeServers
sql://MSSQL$Microsoft##SSEE
http://ppm-itg-mam:2048/ssp/admin
SharePoint - 80
http://ppm-itg-mam/ppm
http://ppm-itg-mam
http://ppm-itg-mam/pwa
Windows SharePoint Services Web Application
Windows SharePoint Services Database
Windows SharePoint Services Search
Windows SharePoint Services coming E-Mail
Windows SharePoint Services Timer
Windows SharePoint Services Administration
Composition
Membership
Containment
Usage
Client-Server

How to Discover Microsoft SharePoint

The following steps describe how to discover Microsoft SharePoint.

1. **Prerequisite - Set up protocol credentials**

   This discovery solution is based on the PowerShell protocol which can also be accessible over NTCMD, SSH, and Telnet protocols at script execution level. Ensure that the corresponding credentials are provided.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Set up user permissions**

The logged in user must have Read permissions on the SharePoint Configuration Database.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** or **Range IPs by nmap** job to discover the SharePoint system IP addresses.

   b. Run the **Host Connection by Shell** or **Host Connection by Powershell** job to discover the connection between SharePoint and the Shell or PowerShell agent, and the networking topology.

   c. Run the **Host Applications by Shell** or **Host Applications by PowerShell** job to discover the connection between the SharePoint system and the SharePoint software element, and the detailed host topology.

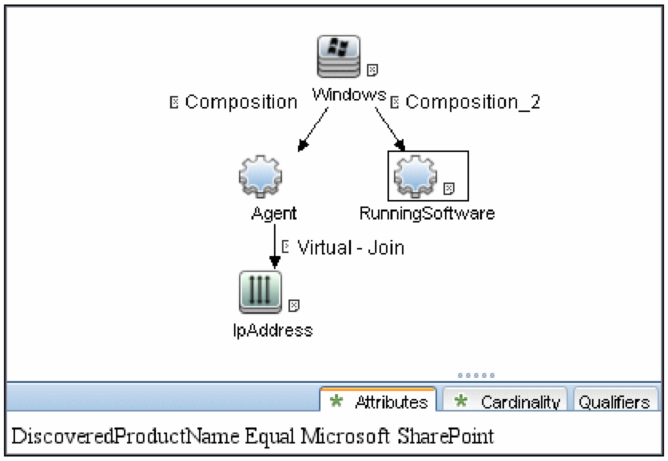
   d. Run the **Microsoft SharePoint Topology** job to discover the Microsoft SharePoint Server topology.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Microsoft SharePoint Topology Job

This section includes details about the job.

Trigger Query

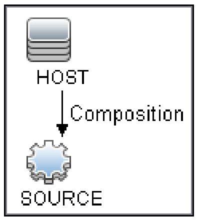


DiscoveredProductName Equal Microsoft SharePoint

Note: On IPAddress, the **IP Probe name is not null** attribute is set.

Adapter

- Input CIT: Agent
- Input Query



- Used Scripts
  - sharepoint_win_shell.py
  - sharepoint.py
  - SharePointMain.py

  Note: This job may also use library scripts supplied with the Auto Discovery Content package.

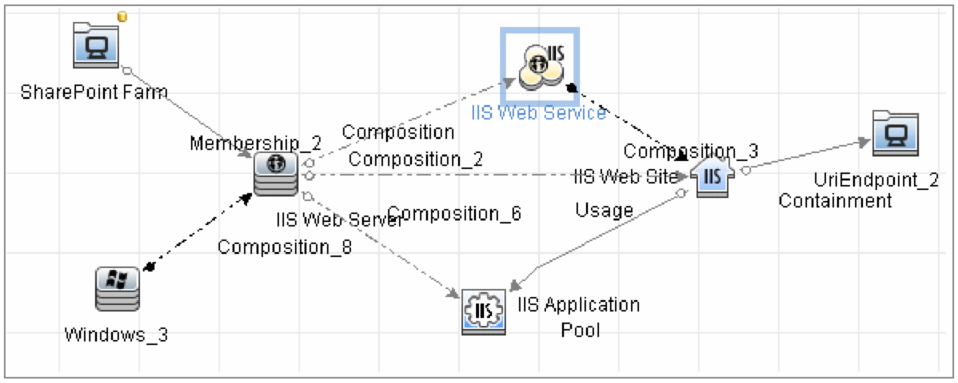
Job Parameters

Parameter	Description
discoverSharePointUrls	Indicates whether or not to discovered URLs of SharePoint sites.
relativeCommandTimeoutMultiplier	The amount of time to wait for the result against the default command execution time.
reportIntermediateWebService	Indicates whether or not the IIS WebService between IIS Web Server and IIS Web Site should be reported. This parameter should be set in accordance with the **report_legacy_topology** parameter of the IIS Application by NTCMD or UDA job.

Depending on the setting of the **reportIntermediateWebService** parameter, this job reports one of the following IIS topologies:

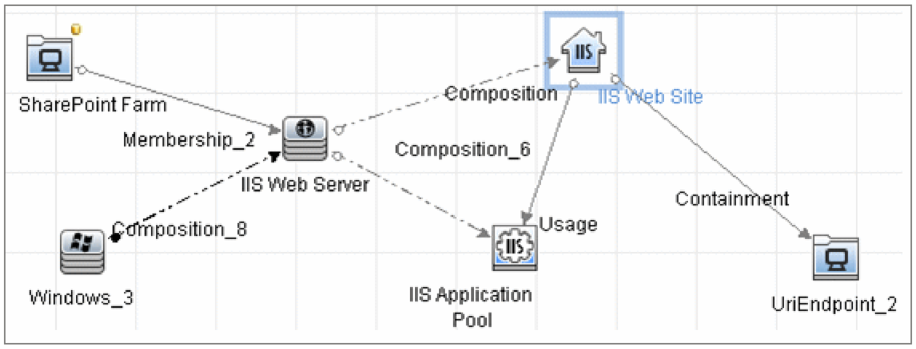
- **reportIntermediateWebService = true**:

  IIS Web Server -> IIS Web Service -> IIS Web Site



- **reportIntermediateWebService = false**:

  IIS Web Server -> IIS Web Service -> IIS Web Site



Created/Changed Entities

Entity Name	Entity Type	Entity Description
sharepoint_farm	CIT	New CIT information regarding the SharePoint farm.
sharepoint_service	CIT	New CIT - a textual file which holds data regarding the SharePoint service configuration
Microsoft SharePoint Topology	Job	New topology job

Entity Name	Entity Type	Entity Description
Application - Microsoft SharePoint	Module	Discovery module
ms_sharepoint_by_shell	Adapter	Discovery adapter
sharepoint_application_agents.xml	TQL query	Trigger TQL query
sharepoint.py	Script	SharePoint topology script
sharepointdiscoverer.py	Script	Script contains mechanism of the SharePoint discovery by Shell and PowerShell
SharePointMain.py	Script	Main script, the job entry point
Sharepoint_xml.ps1	Resource	PowerShell script which represents the SharePoint configuration in XML format

Discovered CITs

- ClientServer
- Composition
- Containment
- IIS Application Pool
- IIS Web Server
- IIS Web Service
- IIS Web Site
- IpAddress
- Membership
- Running Software
- SharePoint Farm

- SharePoint Service
- SQL Server
- UriEndPoint
- Usage
- Windows

Note: To view the topology, see "Topology" on page 577.

Miscrosoft SharePoint Discovery Commands

The SharePoint topology is discovered by running the **Sharepoint_xml.ps1** script. It contains following functions which provide the relevant information in XML format:

This section includes:

ShowSharePointConfig

- **Sample Output**

```
<farm id="4ddfb9c7-754a-4a66-8ee6-7d86613b873c" version="12.0.0.6421">
<hosts> As described for ShowSharePointHostConfig section </hosts>
<webServices> As described for ShowSharePointWebConfig section </webServices>
</farm>
```

- **Modeled CITs: SharePoint Farm**

Attribute	Value
ID	4ddfb9c7-754a-4a66-8ee6-7d86613b873c

ShowSharePointHostConfig

- **Sample Output**

```
<hosts>
  <host name="ucmdb-11">
    <db type="SharedDatabase">Server=ucmdb-11;Database=SharedServices1_DB;Trusted_
Connection=yes;App=Windows SharePoint Services;Timeout=15</db>
    <db type="SPConfigurationDatabase">Server=ucmdb-11;Database=SharePoint_
Config;Trusted_Connection=yes;App=Windows SharePoint Services;Timeout=15</db>
    <service name="Windows SharePoint Services Database">
Databases                    :
NormalizedDataSource         : ucmdb-11
...
    </service>
  </host>
</hosts>
```

- **Modeled CITs: IP**

Attribute	Value
IP Address	Resolved IP of ucmdb-11

- **Modeled CITs: Windows**

Attribute	Value
Host key	'Resolved IP of ucmdb-11' 'IP domain'

- **Modeled CITs: Software Element**

Attribute	Value	Comments
Container	Previously described Windows	
Name	Microsoft SharePoint	
Vendor	microsoft_corp	
Application version	12.0.0.6421	Taken from the SharePoint Farm version attribute

- **Modeled CITs: SQL Server**

Attribute	Value
Container	Previously described windows
Database Name	ucmdb-11
Vendor	microsoft_corp

- **Modeled CITs: SharePoint service**

Attribute	Value
Container	Previously described software element
Name	Windows SharePoint Services Database
Document Data	Databases : NormalizedDataSource : ucmdb-11 ...

ShowSharePointWebConfig

- **Sample Output**

```
<webServices>
  <webService id="c8e64134-0daa-4614-9ed8-257aa653fe9c">
      <applicationPool name="SharePoint - 80">
    <webApplication name="SharePoint - 80">
      <url>http://ddvm-shrpnt/</url>
      <site>http://ddvm-shrpnt</site>
      <site>http://ddvm-shrpnt/personal/administrator</site>
      <site>http://ddvm-shrpnt/ssp/admin</site>
    </webApplication>
  </webService>
</webServices>
```

- **Modeled CITs: Windows**

Attribute	Value
Host key	'Resolved IP of ddvm-shrpnt' 'IP domain'

- **Modeled CITs: IIS**

Attribute	Value	Comments
Container	Previously described Windows	
Name	Microsoft IIS WebServer	
Vendor	microsoft_corp	

- **Modeled CITs: IIS Application Pool**

Attribute	Value
Container	Previously described IIS
Name	SharePoint - 80
Vendor	microsoft_corp

- **Modeled CITs: IIS Website**

Attribute	Value
Container	Previously described IIS
Name	SharePoint - 80

- **Modeled CITs: URL**

Attribute	Value
Container	IIS Host (Windows)
Name	http://ddvm-shrpnt

SharePoint Library Command Flow

The SharePoint library is loaded using the following command flow:

- **[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint");**
- **$spFarm = [Microsoft.SharePoint.Administration.SPFarm]::Local;**
- **if(!$spFarm){echo("---CANNOT EXECUTE DISCOVERY---"); exit(1)}**

After the last command is executed, the local SharePoint farm is initialized or the message **---CANNOT EXECUTE DISCOVERY---** is displayed.

When SharePoint is discovered by PowerShell, the **ShowSharePointHostConfig** and **ShowSharePointWebConfig** commands are called (described in "Miscrosoft SharePoint Discovery Commands" on page 584 above). The SharePoint Farm CI is built from executing the following commands:

- **Echo($spFarm.Id.Guid)** – discovers the farm ID
- **Echo($spFarm.BuildVersion.ToString())** – discovers the farm version

Troubleshooting and Limitations

This section provides troubleshooting and limitations for Microsoft SharePoint discovery.

1. The credential on which the job connects to the SharePoint host must provide a trusted connection to the SharePoint configuration database. If the database host is the third host (discovered host) and the trusted connection is used for the SharePoint configuration database, such configurations will not be discovered. To avoid this problem SQL credentials must be used in the SharePoint configuration.

   The discovery mechanism works in the following cases:

   - The SharePoint configuration database is connected via named pipes (a farm on a single host)
   - An SQL connection is used for the configuration database
   - A trusted connection is used for the configuration database, and this database is hosted with some other SharePoint components

2. For each SharePoint service, all the configuration details are merged into one string in the **service configuration** attribute of the SharePoint Service CIT.

3. If the warning **No SharePoint library found** is displayed, it is recommended to check the Event Viewer on the SharePoint database machine, to see if there are unsuccessful connection attempts from the SharePoint instance which is being discovered. If there are unsuccessful connection attempts, add a new login to MS SQL Server manager (the one which could not access the database) and grant **db_owner** permissions for the **SharePoint_Config** database to this new login.

Introduction to SAP Discovery

SAP Discovery focuses primarily on the SAP architecture layer according to different SAP system configurations. The SAP system configurations supported are:

- ABAP
- JAVA
- DS (Double Stack)

Each SAP system is comprised of:

- NetWeaver Application server, including the following types:
  - Central Instance
  - SCS (Central Services)
  - Dialog Instance
- Message/Enqueue server
- Database
- Components, including:
  - Software Components
  - Development Components

Note: In SAP Discovery, in the case of JAVA and ABAP configurations, one application server is considered as one instance of the configuration, but in the case of DS configuration two application servers with the same instance name on the same host are considered as one instance of the DS configuration.

Discovery Mechanism

SAP Discovery uses two types of mechanisms to perform discovery:

- **Shallow.** SAP Discovery uses application signature and plugins.
- **Deep.** SAP Discovery uses dedicated jobs.

  Most dedicated jobs are separated by the ABAP or JAVA application server type. If a server is identified as part of a DS configuration, then the entire system is marked as DS.

Note: You can discover the entire SAP system by discovering a connection to the SAP Solution Manager. By doing this you create a single set of credentials. There is no need to create a set of credentials for each SAP system.

Application Signatures and Plugins

The following application signatures exist for JAVA and ABAP configurations with the various application servers:

- SAP ABAP SCS
- SAP ABAP Application Server (Dialog)
- SAP ABAP Application Server (Central)
- SAP JAVA SCS
- SAP JAVA Application Server (Dialog)
- SAP Message Server with plugin **plugins_sap_message_server** (relevant for all SAP configurations including ABAP, JAVA, and DS)
- SAP Enqueue Server with plugin **plugins_sap_enqueue_server** (relevant for all SAP configurations including ABAP, JAVA, and DS)

All ABAP-related signatures are processed with the following plugins:

- **plugins_sap_abap_instance_to_system_linkage**
- **sap_cs_and_ap_version_by_shell**
- **sap_cs_and_ap_version_by_wmi**

All JAVA-related signatures are processed with the following plugins:

- **plugins_sap_java_instance_to_system_linkage**
- **sap_cs_and_ap_version_by_shell**
- **sap_cs_and_ap_version_by_wmi**

Note: The following plugins are common to both ABAP-related and JAVA-related signatures:

- **sap_cs_and_ap_version_by_shell**
- **sap_cs_and_ap_version_by_wmi**

Chapter 45: SAP ABAP Discovery

This chapter includes:

Overview

UCMDB discovers the SAP Application Server ABAP, which provides the complete technology and infrastructure to run ABAP applications.

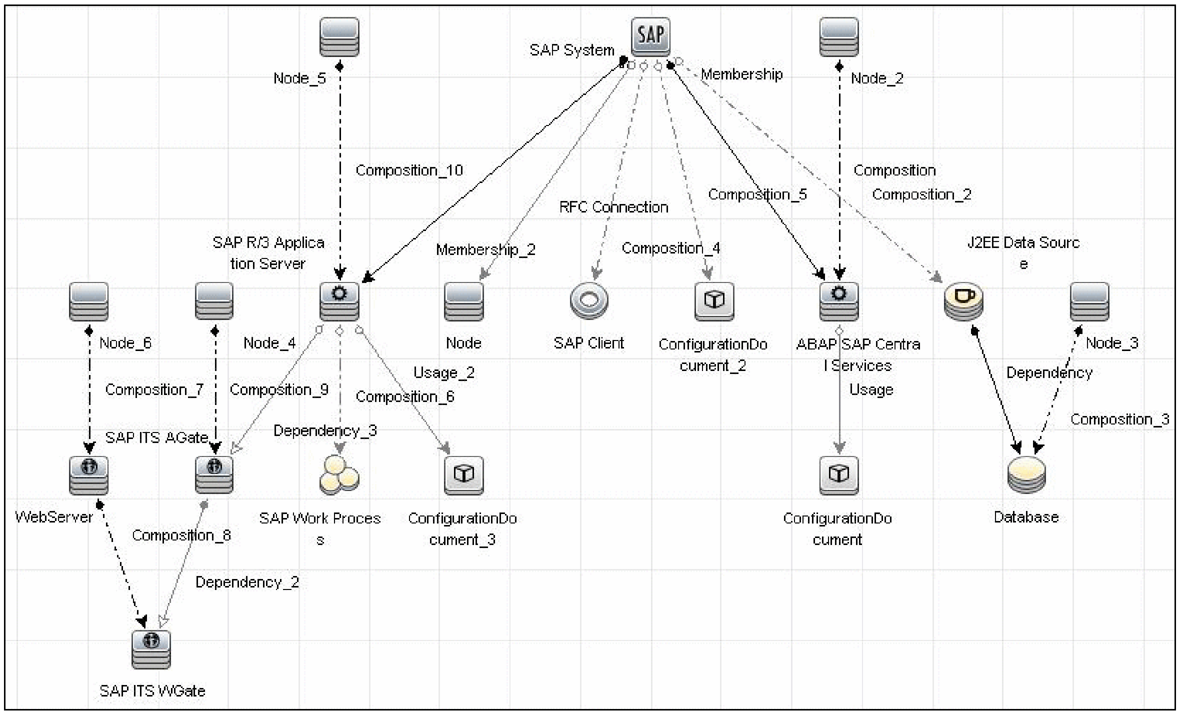
Note: You can discover the whole SAP system by discovering a connection to the SAP Solution Manager. In this way, you create a single set of credentials; there is no need to create a set of credentials for each SAP system. DFM discovers all systems (and their topology) with this one set. For details, see "SAP Solution Manager Discovery" on page 636

Supported Versions

SAP BASIS and SAP AS (Architecture layer)	Versions 3.x to 6.x
SAP JCo.	2.x and 3.x (starting from 3.0.7) Version 3.0.7 and newer is recommended
SAP Solution Manager	Versions 6.x, 7.x

Topology

The following image displays the topology of the SAP ABAP discovery:



How to Discover SAP ABAP

This task discovers SAP ABAP architecture, SAP application components, SAP transactions, and SAP Solution Manager business process definitions. It includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   The following protocols enable connection to a machine to verify whether a SAP system is installed on it:

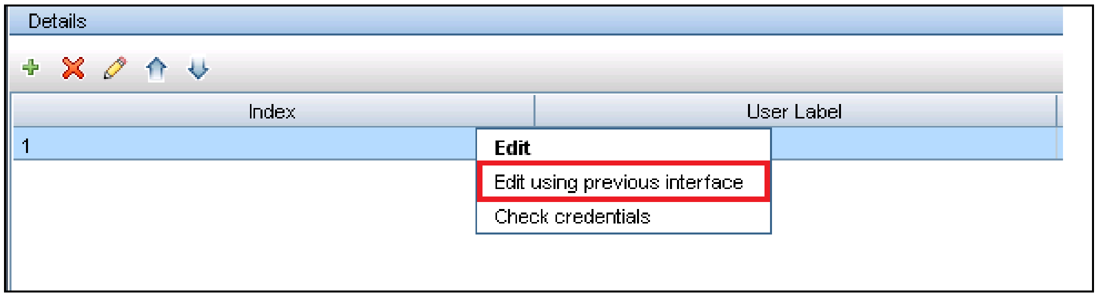
   - NTCMD protocol
   - SSH protocol
   - Telnet protocol
   - SAP protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   To choose correct JCo version:

   a. Create the credentials for the SAP protocol, or choose an existing one.

   b. In the context menu of the created credential, select **Edit using previous interface**, as shown below.



   c. In the dialog window, either confirm that the JCO version is correct, or change it to the correct version. The default version is 2.x.

2. **Prerequisite – Install Java Connectors**

Note: All actions in this part should be performed on the machine where the Data Flow Probe is installed.

JCo version 3.x (from 3.0.7)

a. Download the SAP JCo package. This is accessible from the **SAP Service Marketplace > SAP JCo >Tools & Services** window: http://service.sap.com/connectors

b. Extract the JCo installation ZIP content to a temporary directory (for example: C:\temp).

c. Copy **sapjco3.jar** from the temporary directory to the **<DataFlowProbe_root>\content\lib\** directory

   As an alternative to 2c, you may instead copy **sapjco3.jar** to the **<DataFlowProbe_root>\content\lib\sap\** directory. If you do this, you must change the **<DataFlowProbe_root>\bin\WrapperEnv.conf** file by finding the following string:

   ```
   set.SAP_CLASSES=%sap%/sapjco.jar;%sap%/com_sap_pj_jmx.jar;
   %sap%/exception.jar;%sap%/logging.jar;%sap%/sapj2eeclient.jar;
   %sap%/sapxmltoolkit.jar
   ```

   and adding **;%sap%/sapjco3.jar** to the end like this:

   ```
   set.SAP_CLASSES=%sap%/sapjco.jar;%sap%/com_sap_pj_jmx.jar;
   %sap%/exception.jar;%sap%/logging.jar;%sap%/sapj2eeclient.jar;
   %sap%/sapxmltoolkit.jar;%sap%/sapjco3.jar
   ```

d. Copy **sapjco3.dll** from the temporary directory to the DataFlowProbe directory containing the file sapjco3.jar.

e. In the **WrapperEnv.conf** file on the probe, change the Environment Discovery Path for the SAP CLASSES by replacing **%sap%/sapjco.jar** with **%sap%/sapjco3.jar**.

JCo version 2.x

a. Download the SAP JCo package. This is accessible from the **SAP Service Marketplace > SAP JCo >Tools & Services** window: http://service.sap.com/connectors

b. Extract the JCo installation ZIP content to a temporary directory (for example: C:\temp).

c. Copy **sapjco.jar** from the temporary directory to the **<DataFlowProbe_root>\content\lib\** directory

As an alternative to 2c, you may instead copy **sapjco.jar** to the **<DataFlowProbe_root>\content\lib\sap\** directory. If you do this, you must ensure that **sapjco.jar** is defined in the SAP_CLASSES parameter of the **<DataFlowProbe_root>\bin\WrapperEnv.conf** file as shown in the following string:

```
set.SAP_CLASSES=%sap%/sapjco.jar;%sap%/com_sap_pj_jmx.jar;
%sap%/exception.jar;%sap%/logging.jar;%sap%/sapj2eeclient.jar;
%sap%/sapxmltoolkit.jar
```

d. Copy **sapjcorfc.dll** from the temporary directory to the DataFlowProbe directory containing the file sapjco.jar.

e. Copy **librfc32.dll** from the temporary directory to the directory for the shared libraries where it can be loaded by linker. This is usually the **%winnt%** or **%winnt%\System32\** directory. See the JCo README for details.

3. **Configure adapter parameters**

To specify exactly which CIs to discover, or to omit unnecessary CIs, you can configure the adapter parameters, as follows:

Discovery	Configuration
To discover all SAP transactions	Set **getAllTransactions** to **true**
To discover active SAP transactions	Set **getActiveTransactions** to **true**
To discover SAP transactions that were changed by discovered transports	▪ Set **getTransChanges** to **true** ▪ Set the **from** date (**transChangesFromDate**) and the **to** date (**transChangesToDate**). The date format is **MM/DD/YYYY** or **YYYYMMDD**. ▪ Set the **from** time (**transChangesFromTime**) and the **to** time (**transChangesToTime**). The time format is **HH:MM:SS** or **HHMMSS**.

For details on configuring adapter parameters, see the section describing Adapter Management in the *HP Universal CMDB Data Flow Management Guide*.

4. Run the discovery

a. In the Universal Discovery window, activate the jobs in the following order:

For details on running jobs, see the section describing the Module/Job-Based Discovery in the *HP Universal CMDB Data Flow Management Guide*

- **Range IPs by ICMP** or **Range IPs by nmap**, **Host Connection By Shell**.
- **Host Resources and Applications by Shell.** Discovers SAP running software and processes.
- **SAP TCP Ports.**
- **SAP System by Shell.** Searches for an SAP system by referring to the file system and process list. The SAP CI that is created is used as a trigger for the **SAP ABAP Connection by SAP JCO** job. This job needs Shell credentials and not SAP credentials.
- **SAP ABAP Connection by SAP JCO.** Connects to the SAP system and creates a SAP System CI with a credentials ID. Subsequently, the other ABAP jobs use these credentials to connect to SAP.
- **SAP ABAP Topology by SAP JCO.** Discovers infrastructure entities in the SAP system: hosts, application servers, work processes, databases, SAP clients, configuration files, software components (discovered as configuration files), and support packages (discovered as configuration files).
- **SAP Applications by SAP JCO.** Discover the application components of this system. The result of this job may be many CIs. To omit unnecessary CIs, you can configure the adapter parameters. For details, see "Configure adapter parameters" on the previous page.
- **SAP ITS by NTCMD or UDA.** Discovers Internet Transaction Server (ITS) entities (Application Gateway and Web Gateway).
- **SAP Solution Manager by SAP JCO.** Discovers SAP Solution Manager components. SAP Solution Manager discovery enables you to discover the business process hierarchy. For details, see "SAP Solution Manager Discovery" on page 636.

b. For details on the CIs that are discovered, see the section describing the Discovery Job Details Pane in the *HP Universal CMDB Data Flow Management Guide*.

c. Verify that DFM discovered the appropriate components. Access the **SAP_ABAP_Topology** view in the Modeling Studio and verify that the map displays all components.

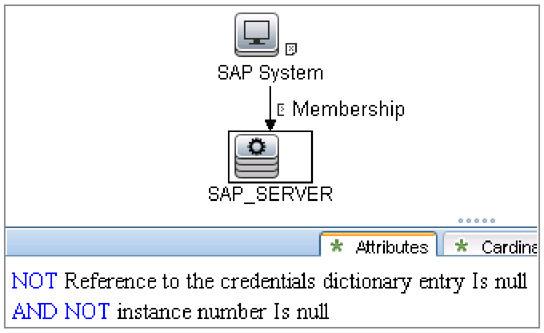
d. To view the CIs discovered by the SAP APAB discovery, see the section describing the Discovered CIs Window in the *HP Universal CMDB Data Flow Management Guide*.

SAP Solution Manager Topology by SAP JCO Job

This section includes details about the job.

Trigger Query

Trigger CI: SAP ABAP Application Server



NOT Reference to the credentials dictionary entry Is null
AND NOT instance number Is null

Used Scripts

- db.py
- db_builder.py
- db_platform.py
- iteratortools.py
- jdbc.py, jmx.py
- jdbc_url_parser.py
- jee.py
- sap.py
- sap_abap.py
- sap_abap_discoverer.py
- sap_discoverer.py
- sap_jee.py

- sap_solman_discoverer.py
- sap_solman_topology.py
- saputils.py

Discovered CITs

- ABAP SAP Central Services
- Composition
- Configuration Document
- Containment
- Database
- Dependency
- IpAddress
- J2EE Cluster
- J2EE SAP Central Services
- JDBC Data Source
- Membership
- Node
- SAP ABAP Application Server
- SAP Client
- SAP J2EE Application Server
- SAP System
- Usage

Adapter Parameters

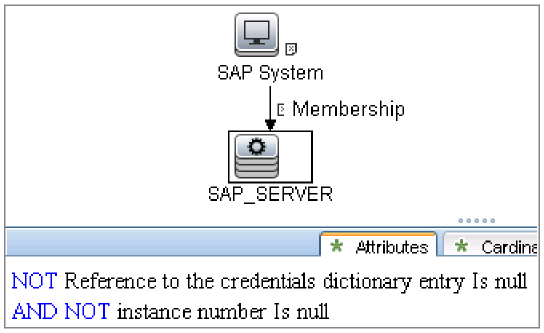
- **remoteJVMArgs.** The JVM parameters that should be passed to the remote process.
- **runInSeparateProcess.** If true, the pattern runs in a separate thread.

SAP Solution Manager by SAP JCO Job

This section includes details about the job.

Trigger Query

- Trigger CI: SAP ABAP Application Server
- Trigger query:



NOT Reference to the credentials dictionary entry Is null
AND NOT instance number Is null

Used Scripts

- cmdlineutils.py
- command.py
- iteratortools.py
- sap.py
- sap_abap.py
- sap_abap_discoverer.py
- sap_discoverer.py
- sap_solman_discoverer.py
- sap_solution_manager.py
- saputils.py

Discovered CITs

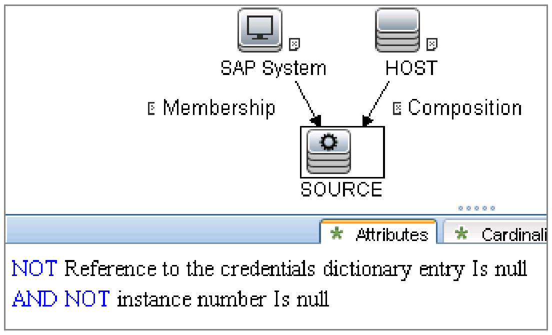
- Composition
- Containment
- IpAddress
- Membership
- Node
- SAP ABAP Application Server
- SAP Business Process
- SAP Business Scenario
- SAP Process Step
- SAP Project
- SAP System
- SAP Transaction

SAP Applications by SAP JCO Job

This section includes details about the job.

Trigger Query

- Trigger CI: SAP ABAP Application Server
- Trigger query:



NOT Reference to the credentials dictionary entry Is null
AND NOT instance number Is null

Used Scripts

- sapapputils.py
- saputils.py
- sap_applications.py

Discovered CITs

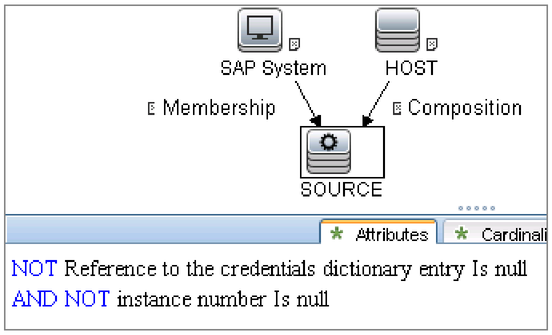
- Composition
- Containment
- SAP Application Component
- SAP System
- SAP Transaction
- SAP Transport
- SAP Transport Change
- Usage

SAP ABAP Topology by SAP JCO Job

This section includes details about the job.

Trigger Query

- Trigger CI: SAP ABAP Application Server
- Trigger query:



NOT Reference to the credentials dictionary entry Is null
AND NOT instance number Is null

Used Scripts

- cmdlineutils.py
- command.py
- db.py
- db_builder.py
- db_platform.py
- entity.py
- iteratortools.py
- sap.py
- sap_abap.py
- sap_abap_discoverer.py
- sap_discoverer.py
- sap_flow.py

- sap_site_by_jco.py
- saputils.py

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database
- Dependency
- IPAddress
- JDBC Data Source
- Membership
- Node
- RFC Connection
- RunningSoftware
- SAP ABAP Application Server
- SAP Client
- SAP Gateway
- SAP System
- SAP Work Process
- Usage

Adapter Parameters

- **discoverRFCConnections.** If **true**, this job gets the RFC Connections of the SAP system.

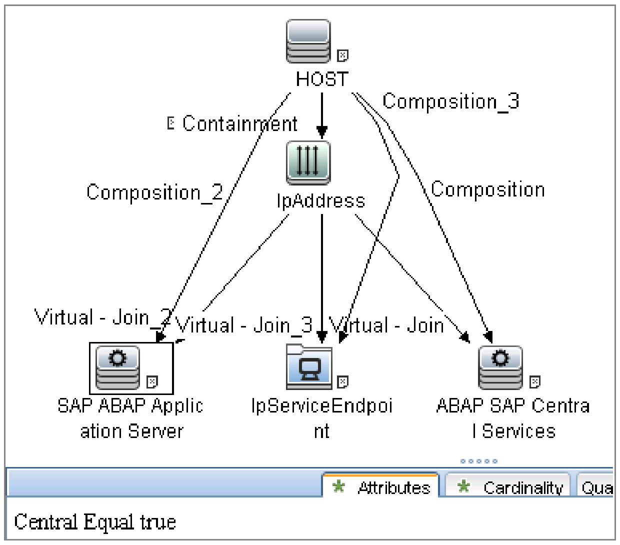
- **discoverSAPProfiles.** If **true**, this job discovers SAP profiles by querying SAP database.
- **remoteJVMArgs.** The JVM parameters that should be passed to the remote process.
- **reportComponentsAsConfigFile.** If **true**, this job reports software components as a registry (configuration file). If **false**, this job reports separate CIs per component. The default value is **true**.
- **runInSeparateProcess.** If true, the pattern runs in a separate thread.

SAP ABAP Connection by SAP JCO Job

This section includes details about the job.

Trigger Query

- Trigger CI: IpAddress
- Trigger query:



Central Equal true

Used Scripts

- saputils.py
- sap_system_dis.py

- iteratortools.py
- cmdlineutils.py
- entity.py
- command.py
- sap_flow.py
- sap.py
- sap_abap.py
- sap_discoverer.py
- sap_abap_discoverer.py

Discovered CITs

- Composition
- Containment
- IPAddress
- Membership
- Node
- SAP ABAP Application Server
- SAP System

Adapter Parameters

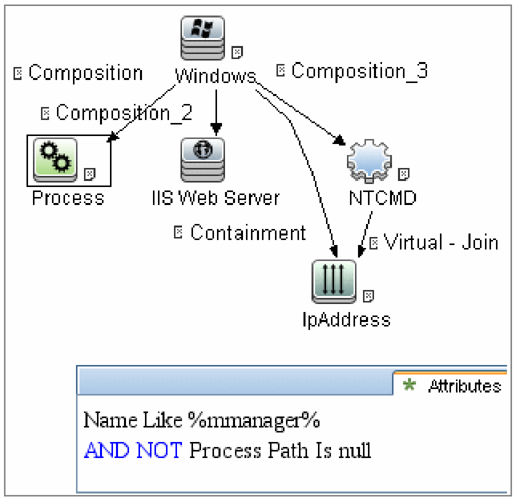
- **remoteJVMArgs.** The JVM parameters that should be passed to the remote process.
- **runInSeparateProcess.** If true, the pattern runs in a separate thread.

SAP ITS by NTCMD or UDA Job

This section includes details about the job.

Trigger Query

- Trigger CI: IIS Web Server
- Trigger query:



Name Like %mmanager%
AND NOT Process Path Is null

Used Script

- cmdlineutils.py
- command.py
- entity.py
- iteratortools.py
- sap.py
- sap_abap.py
- sap_its.py
- saputils.py

Discovered CITs

- Composition

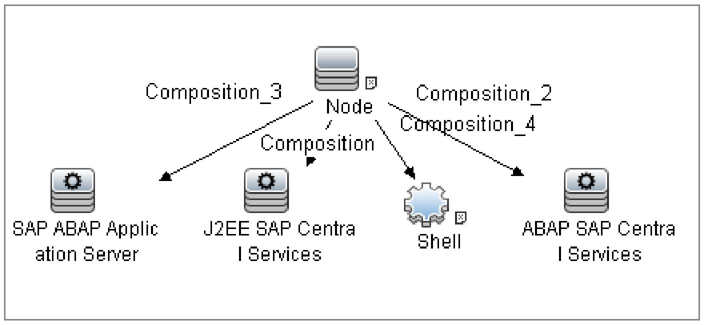
- Containment
- Dependency
- IPAddress
- Node
- SAP ABAP Application Server
- SAP ITS AGate
- SAP ITS WGate
- WebServer

SAP System by Shell Job

This section includes details about the job.

Trigger Query

- Trigger CI: SapApplicationServer
- Trigger query:


Node
Composition_3
Composition
Composition_2
Composition_4
SAP ABAP Application Server
J2EE SAP Central Services
Shell
ABAP SAP Central Services

Used Scripts

- cmdlineutils.py
- entity.py

- sap.py
- sap_discoverer.py
- sap_system_by_shell.py

Discovered CITs

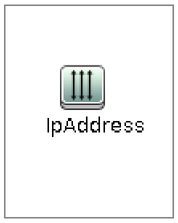
- Composition
- ConfigurationDocument
- Membership
- SapApplicationServer
- SAP System
- Usage

SAP TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host’s ports.
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).

Name	Default Value	Description
ports	**For JEE TCP Ports job:** • weblogic, weblogicSSL, websphere_jmx, rmi **For Database TCP Ports job:** • oracle, db2, sybase, sql, mysql **For SAP TCP Ports job:** • sap, sap_jmx, sap_http, sap_https **For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.

Name	Default Value	Description
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

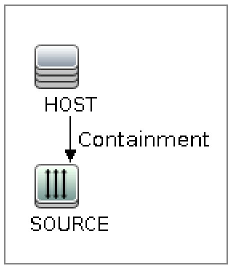
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

Troubleshooting and Limitations

- **Problem:** The SAP discovery fails and a Java message is displayed:

  ```
  This application has failed to start because MSVCR71.dll was not found.
  ```

  Solution: Two .dll files are missing. For the solution, read Note #684106 in https://websmp205.sap-ag.de/~form/sapnet?_FRAME=CONTAINER&_OBJECT=012003146900000245872003.

- **Problem:** The SAP ABAP discovery job fails with error "SAP drivers are missing", even if SAP Java Connector drivers are installed.

  Solution 1: The Discovery Probe is trying by default to connect using JCo 3 drivers, but these drivers are not installed. Therefore, install JCo 3.x drivers.

  Solution 2: The Discovery Probe is trying by default to connect using JCo 3 drivers, but the SAP system does not support JCo 3. For the solution, go to **Data Flow Probe Setup** and right-click on the required permission in **SAP Protocol**. Select **Edit using previous interface**, change **JCo version** to **2.x**, even if it is already selected, and save the permission.

- **Problem:** You experience difficulties in loading the SAP JCo dll files.

  Solution: Check you have the latest version of the Microsoft Visual Studio C/C++ runtime redistributable package installed.

Chapter 46: SAP Java Discovery

This chapter includes:

Overview

UCMDB discovers the SAP Application Java Server, which provides a Java 2 Enterprise Edition (J2EE) environment for developing and running Java Enterprise Edition programs.

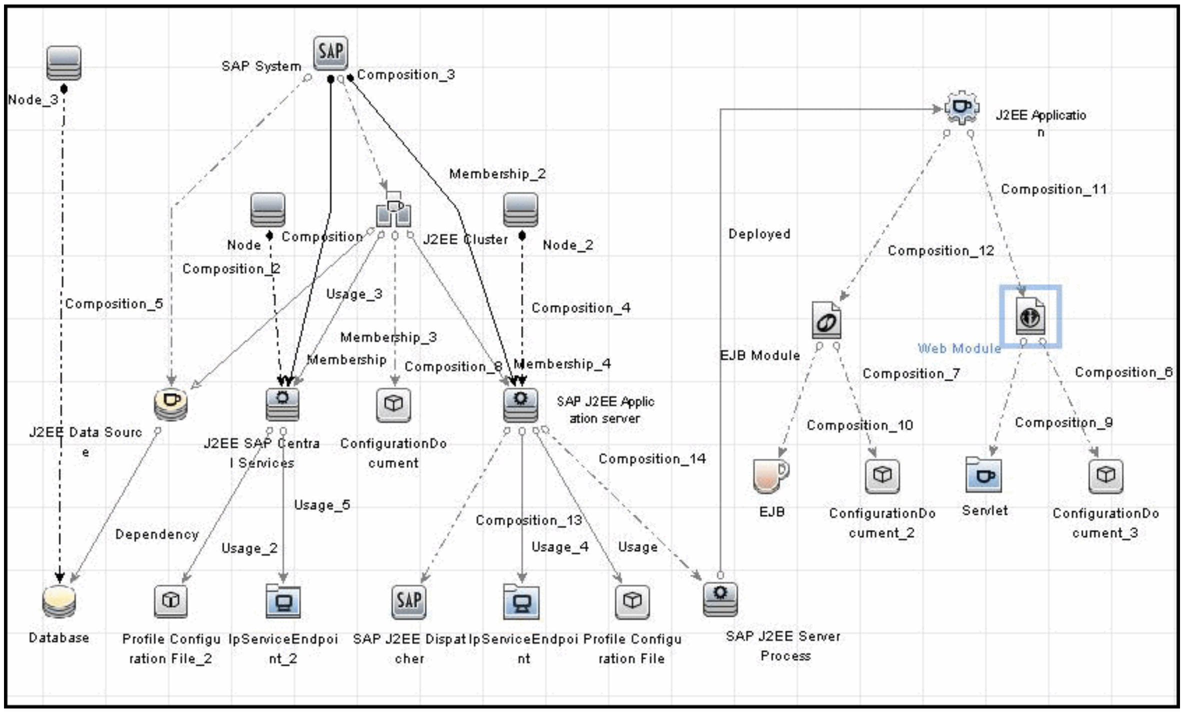
Note: You can discover the whole SAP system by discovering a connection to the SAP Solution Manager. In this way, you create a single set of credentials; and there is no need to create a set of credentials for each SAP system. Universal Discovery discovers all systems (and their topology) with this one set. For details, see "SAP Solution Manager Discovery" on page 636.

Supported Versions

SAP BASIS and SAP AS (Architecture layer)	Versions 3.x to 6.x
SAP J2EE client	The version should match the relevant SAP system version
SAP Solution Manager	Versions 6.x, 7.x

Topology

The following diagram depicts the elements in the SAP Java topology.

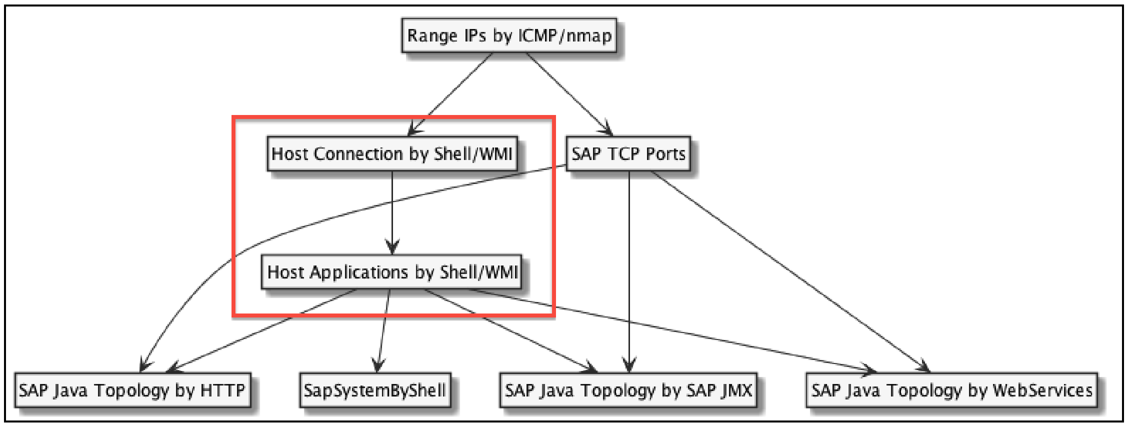

SAP System
Composition_3
Node_3
J2EE Application
Membership_2
Composition_11
Node
Composition
J2EE Cluster
Node_2
Deployed
Composition_2
Composition_12
Usage_3
Composition_5
Composition_4
Membership_3
EJB Module
Web Module
Membership
Composition_8
Membership_4
Composition_7
Composition_6
SAP J2EE Application server
J2EE Data Source
J2EE SAP Central Services
ConfigurationDocument
Composition_10
Composition_9
Composition_14
Usage_5
EJB
ConfigurationDocument_2
Servlet
ConfigurationDocument_3
Dependency
Composition_13
Usage_2
Usage_4
Usage
Database
Profile Configuration File_2
IpServiceEndpoint_2
SAP J2EE Dispatcher
IpServiceEndpoint
Profile Configuration File
SAP J2EE Server Process

How to Discover Full SAP Java-related Topology

The SAP Java Discovery enables you to discover the full SAP Java-related topology and J2EE applications on the SAP JAVA server.

The jobs that you need to run to perform this discovery are determined by whether shell access is provided to SAP destinations. If shell access is provided, then perform discovery as described in "SAP Java Discovery with Shell Access Provided to SAP Destinations" below. If shell access is not provided to SAP destinations, perform discovery as described in "SAP Java Discovery without Shell Access Provided to SAP Destinations" on the next page.

The following diagram depicts the various scenarios for running SAP Java Discovery.


Range IPs by ICMP/nmap
Host Connection by Shell/WMI
SAP TCP Ports
Host Applications by Shell/WMI
SAP Java Topology by HTTP
SapSystemByShell
SAP Java Topology by SAP JMX
SAP Java Topology by WebServices

SAP Java Discovery with Shell Access Provided to SAP Destinations

Note: When shell access is provided to SAP destinations, SAP Java Discovery can be performed either in shallow mode or in deep mode. To perform a shallow discovery, carry out steps 1-3 below. To perform a deep discovery, carry out all of the steps below.

To perform the SAP Java Discovery:

1. In the Universal Discovery window, run the **Range IPs by ICMP** job. This job discovers all reachable IPs.

2. Run the **Host Connection By Shell** job. This job discovers the host's connectivity by shell protocol to application servers.

3. Run the **Host Applications by Shell** job. This job discovers application servers, message and

enqueue servers, and their relationships to the SAP system.

> **Note:** If you want to perform only a shallow discovery, stop here and do not continue to the next step.

4. Run one or more of the following jobs(depending on the type of Java topology you want to discover):

   - **SAP Java Topology by HTTP.** This job discovers the full SAP Java topology, including databases used. It determines system configuration, which is either standalone Java or Double Stack.
   - **SAP Java Topology by SAP JMX.** This job discovers the full SAP Java topology with details about instances. Databases are discovered only when the new MBean model is available (starting from Java application server version 7.1). It is not possible to determine the type of system configuration.
   - **SAP Java Topology by WebServices.** This job is the same as the JMX job, but with a different transport.
   - **SAP System by Shell**

SAP Java Discovery without Shell Access Provided to SAP Destinations

To perform the SAP Java Discovery:

1. In the Universal Discovery window, run the **Range IPs by ICMP** job. This job discovers all reachable IPs.
2. Run the job **SAP TCP Ports** job.
3. Run one or more of the following jobs (depending on the type of Java topology you want to discover):

   - **SAP Java Topology by HTTP.** This job discovers the full SAP Java topology, including databases used. It determines system configuration, which is either standalone Java or Double Stack.
   - **SAP Java Topology by SAP JMX.** This job discovers the full SAP Java topology with details about instances. Databases are discovered only when the new MBean model is available (starting from Java application server version 7.1). It is not possible to determine the type of system configuration.

- **SAP Java Topology by WebServices.** This job is the same as the **SAP Java Topology by SAP JMX.** job, but with a different transport.

SAP Java Topology by HTTP Job

This job is based on parsing data from the XML queried HTTP URLs. URLs can differ slightly depending on the system configuration where the application server (instance) resides. There are two main types of configurations that are used by this job:

1. **Pure/standalone Java system configuration**, which has the following type of query:
   - **http(s)://<address>:<port>/sap/monitoring/SystemInfoServlet** (used to query information about the entire system)
2. **Double Stack (DS) configuration**, which has the following type of query:
   - **http(s)://<address>:<port>/monitoring/SystemInfoServlet** (used to query information about the entire system)

Adapter

ID: `sap_java_topology_by_http`

Trigger TQL

This job is triggered on IPs that are part of the host with endpoint `sap_http` or `sap_https`.

Parameters

None

Prerequisites

1. **Set up credentials.** The HTTP client uses the same credentials that are used for the **SAP Java Topology by SAP JMX** job.
2. **Libraries installation on the probe.** There is no need to install external libraries. All required dependencies come with a Probe installation.

Discovery Flow

The discovery flow for the SAP Java Topology by HTTP Job is as follows:

1. Get available SAP JMX credentials for this destination.

   Note: Even if the port in specified in the credentials is different from destination port, such credential are also used.

2. For each credential, this job attempts to perform an HTTP query using the **SystemInfoServlet** URLs mentioned above. Successful output is XML that is parsed to get the following SAP system details:

   - system ID (from the XML: **/SAP_J2EE/SID/@value**)
   - system installation type (from the XML: **/SAP_J2EE/INSTALLATION_TYPE/@value**)
   - version, which is composed of:
     - **/SAP_J2EE/REL/@value**
     - **/SAP_J2EE/PATCH_LEVEL/@value**
   - database information (from the XML: **/SAP_J2EE/DB_INSTANCE**)
   - SCS (from the XML: **/SAP_J2EE/SCS_INSTANCE**)

     Note: SCS data is not complete because it contains only port and host information, while the instance number is missing. Therefore, this job computes the instance number from the port. Several port patterns considered, such as 3xNN, 81NN (http), 444NN (https), where NN stands for instance number.

   - dialog instances (from the XML: **/SAP_J2EE/DIALOG_INSTANCE**)

     Note: **/SAP_J2EE/DIALOG_INSTANCE/INSTANCE_DIR** and **/SAP_J2EE/DIALOG_ INSTANCE/NAME** are intentionally skipped from the discovery algorithm as they may contain invalid data.

   - central instance (from the XML: **/SAP_J2EE/CENTRAL_INSTANCE**)

     Reporting dialog and central instances contains some additional logic for instance name resolving. To get the instance name, this job collects information on all workers (or servers) and a dispatcher to get runtime properties such as **application.home**, **com.sap.jvmdir**, **java.home**, **rdbms.driverLocation**, and **user.dir**. This job attempts to find a path with the corresponding SID and with a valid instance name from the paths that have been discovered thus far. Usually **application.home** contains such information.

- software components (from the XML: **/SAP_J2EE/SOFTWARE_COMPONENTS/COMPONENT**)

SAP Java Topology by SAP JMX Job

This job uses the SAP Java client to access the MBean server of an application server via the P4 port (5xx04).

> **Note:** The P4 port is usually closed and starting from Java application server version 7.x it is recommended to use the **SAP Java Topology by WebServices** and/or **SAP Java Topology by HTTP** jobs.

Adapter

ID: `SAP_Dis_J2EE_Site`

Trigger TQL

This job is triggered on IPs that are part of the host with endpoint `sap_jmx`.

Parameters

None

Prerequisites

1. **Set up credentials.**

   To set up credentials to be used by the Java client discovery, you must provide separate credentials called **SAP JMX**. The **SAP JMX** credentials enable connection to a machine and verification of whether a SAP system is installed on it.

2. **Add java client (jar files) to DataFlowProbe machine.**

   > **Note:** If you create version folders under the **\j2ee\sap** directory on the Data Flow Probe machine, you can connect to several SAP versions by adding **.jar** files to each folder. For example, to connect to versions SAP 6.4 and 7.0, in the SAP folder create two sub-folders called 6.x and 7.x and place the relevant **.jar** files in these folders.

   a. Add the following **.jar** files to the **<DataFlowProbe_root>\runtime\probeManager\discoveryResources\j2ee\sap** directory on the Data Flow Probe machine:

- sapj2eeclient.jar
- logging.jar
- exception.jar
- sapxmltoolkit.jar

The files reside in the **\usr\sap\<SID>\<instance name>\j2ee\j2eeclient** directory on one of the SAP instance machines.

b. Add the **com_sap_pj_jmx.jar** file to the **<DataFlowProbe_root>\runtime\probeManager\discoveryResources\j2ee\sap** directory on the Data Flow Probe machine.

The file resides in the **\usr\sap\<SID>\<instance name>\j2ee\admin\lib** directory on one of the SAP instance machines.

Discovery Flow

1. Get available credentials for this destination and filter them so that only credentials that do not have a port or with a port that belongs to the list of ports opened on a destination are used.

2. For each credential, this job attempts to establish a connection to the destination.

3. Once the connection is established, this job sends the query **SAP_J2EECluster** to get cluster details. This query returns the following information:

   - **Name.** This attribute contains the name of the cluster/system.
   - **InstanceNames.** This attribute contains the names of the Java application servers available in this system.

   After this query initial topology is reported.

4. To get more details about each instance, this job repeats the **SAP_J2EECluster**query to the same MBean but with the additional attribute `AllInstanceInfos`. The hostname and instance number are determined based on the `Name` value.

5. If the parameter **reportComponentsAsConfigFile** is set to **true**, this job discovers Development

Components. The following MBeans are used to query interfaces, libraries, and services respectively:

- `SAP_J2EEInterfacePerNode`
- `SAP_J2EELibraryPerNode`
- `SAP_J2EEServicePerNode`

6. This job discovers SCS (Central Services) application servers using the query `SAP_J2EEInstance` with `name` set to `SCS`.

7. This job discovers information about one or more workers and a dispatcher using the query `SAP_J2EEClusterNode`.

8. This job attempts to discover databases using the query `SAP_ITSAMJ2eeCluster`. However, this works only for the new MBean model (starting from version 7.1).

SAP Java Topology by WebServices Job

This job is based on the same MBean model as the job **SAP Java Topology by SAP JMX**, but uses the SAP WebServices transport. This job also shares the same code base as the **SAP Java Topology by SAP JMX** job, and slightly differs in the discovery flow due to a deserialization limitation.

Adapter

ID: `SAP_Dis_J2EE_Site`

Trigger TQL

This job is triggered on IPs that are part of the host with endpoint **sap_http**.

Parameters

- **remoteJVMClasspath.** This parameter specifies additional libraries in the **classpath** of the remote process, where the **wsconnector** library is required. The default value is **%minimal_classpath%;../lib/wsconnector.jar**.

Prerequisites

1. **Set up credentials**

   The WebServices client uses the same credentials that are used for the **SAP Java Topology by SAP JMX** job.

2. **Libraries installation on the probe**

   There is no need to install external libraries. All required dependencies come with a Probe installation.

Discovery Flow

The flow for this job is very similar to the flow of the **SAP Java Topology by SAP JMX** job, but there are some differences when discovering whole system information.

Note: Due to the deserialization limitation some discovered instances are reported without **instance_name**, as this information is not available.

The discovery flow for the SAP Java Topology by WebServices job is as follows:

1. Gets available credentials for this destination and filter them so that only credentials that do not have a port or have a port that belongs to the list of ports opened on a destination are used.

2. For each credential, this job attempts to establish a connection to the destination using WebServices.

3. Once the connection is established, this job discovers all application servers by their names by sending the query `SAP_J2EECluster` to get cluster details. This query returns the following information:

   - **Name.** This attribute contains the name of the cluster/system.

   - **InstanceNames.** This attribute contains the names of the Java application servers available in this system.

   After this query initial topology is reported.

   Note: At this point, this job misses discovery of the instance details due to the deserialization limitations of the WebServices client.

4. If the parameter **reportComponentsAsConfigFile** is set to **true**, this job discovers Development Components. The following MBeans are used to query interfaces, libraries, and services respectively:
   - **SAP_J2EEInterfacePerNode**
   - **SAP_J2EELibraryPerNode**
   - **SAP_J2EEServicePerNode**
5. This job discovers SCS (Central Services) application servers using the query **SAP_J2EEInstance** with **name** set to **SCS**.
6. This job discovers information about one or more workers and a dispatcher using the query **SAP_J2EEClusterNode**.
7. This job attempts to discover databases using the query **SAP_ITSAMJ2eeCluster**. However, this works only for the new MBean model (starting from version 7.1).

SAP Java Topology by HTTP Adapter

ID

`sap_java_topology_by_http`

Input CIT

`IpServiceEndpoint`

Input TQL

The SAP Java Topology by HTTP job is triggered when `IpServiceEndpoint` has the name `sap_http` or `sap_https`.

Triggered CI Data

- **ip_service_name.** The name of the IP service endpoint.

- **ip_port_pai.** The string representation of the endpoint, similar to **address:port**.

Used Scripts

The entry point module is `sap_jee_topology_by_http.py`.

Discovered CITs

- Database
- IpAddress
- IpServiceEndpoint
- J2EE Cluster
- SapJ2eeApplicationServer
- SapJ2eeCentralServices
- SapJ2eeDispatcher
- SapJ2eeServerProcess
- SapJavaSoftwareComponent
- SapSystem

Parameters

None

SAP Java Topology by JMX Adapter

ID

`SAP_Dis_J2EE_Site`

Input CIT

`IpAddress`

Input TQL

The SAP Java Topology by JMX job is triggered when `IpAddress` belongs to the host and `IpServiceEndpoint` exists with the same address. The name of `IpServiceEndpoint` must be or `sap_jmx` or `sap_http`.

Triggered CI Data

- **hostId.** The destination UCMDB ID.
- **ip_address.** The destination IP address.
- **ip_domain.** The destination domain.
- **sap_jmx_port.** The optional port of the endpoint.

Used Scripts

The entry point module is `sap_jee_topology_by_jmx.py`.

Discovered CITs

- Composition
- Configuration Document
- Containment
- Database
- Dependency
- Deployed
- EJB Module
- EJB
- Entity Bean
- IpAddress

- IpServiceEndpoint
- J2EE Cluster
- J2EE Domain
- J2eeApplication
- JDBC Data Source
- Membership
- Message Driven Bean
- Node
- RunningSoftware
- SapJ2eeApplicationServer
- SapJ2eeCentralServices
- SapJ2eeDispatcher
- SapJ2eeServerProcess
- SapJavaSystemComponents
- SapSystem
- Servlet
- Stateful Session Bean
- Stateless Session Bean
- Usage
- Web Module

Parameters

The following two parameters declare how to run jobs defined on top of this adapter:

- **runInSeparateProcess.** When this parameter is set to `true`, this job runs in a separate process. The default value is `true`.
- **remoteJVMArgs.** Contains JVM parameters that should be passed to the remote process.

The following parameter influences topology reporting flow:

- **reportComponentsAsConfigFile.** When set to `true`, the job reports Java Development Components as a configuration document. When set to `false`, the job reports separate CIs per component. The default value is `true`.

Troubleshooting and Limitations

If you complete all prerequisites, but the discovery returns a “Connection Failed” message, review **RemoteProcesses.log** in the DDM Flow Probe logs folder(**C:\hp\UCMDB\DataFlowProbe\runtime\log**). If "NoClassDefFoundError" is displayed there, use the following workaround:

1. Copy the following SAP jar files to the **C:\hp\UCMDB\DataFlowProbe\content\lib\sap** folder:
   - sapj2eeclient.jar
   - logging.jar
   - exception.jar
   - sapxmltoolkit.jar
   - com_sap_pj_jmx.jar

   If the sap folder does not exist, create it.
2. Restart the Data Flow Probe.

If you use this workaround, you may only use one version of SAP jar files.

Chapter 47: SAP Solution Manager Discovery

This chapter includes:

Overview

Often, an environment includes more than one SAP system, each one using a different set of credentials (for instance, user name, password, system number, or client number).

It is customary to register all SAP systems in the SAP Solution Manager, to centralize the management of the SAP systems. DFM enables discovery of all the SAP systems by discovering this connection to the SAP Solution Manager. In this way, you create a single set of credentials; there is no need to create a set of credentials for each SAP system. DFM discovers all systems (and their topology) with this one set.

Supported Versions

SAP BASIS and SAP AS (Architecture layer)	Versions 3.x to 6.x.
SAP JCo.	2.x and 3.x (starting from 3.0.7) Version 3.0.7 and newer is recommended
SAP Solution Manager	Versions 6.x, 7.x.

Topology

To view the SAP Solution Manager Topology by SAP JCO topology: **Universal Discovery** > select **Enterprise Applications > SAP > SAP Solution Manager Topology by SAP JCO > Details pane**. Click the **View CIs in Map** button.

How to Discover SAP Solution Manager

DFM discovers the SAP business layer and the complete topology of registererd SAP systems. It includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery solution is based on the SAP protocol**.**

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Set up permissions**

   To run SAP Solution Manager, ask the SAP Solution Manager administrator to give you permissions on the following objects for the given profile:

   - For the **S_RFC** object, obtain privileges: RFC1, SALX, SBDC, SDIF, SDIFRUNTIME, SDTX, SLST, SRFC, STUB, STUD, SUTL, SXMB, SXMI, SYST, SYSU, SEU_COMPONENT.
   - For the **S_XMI_PROD** object, obtain:

     ```
     EXTCOMPANY=MERCURY;EXTPRODUCT=DARM;INTERFACE=XAL
     ```

   - For the **S_TABU_DIS** object, obtain:

     ```
     DICBERCLS=SS; DICBERCLS=SC; DICBERCLS=&NC& ACTVT=03
     ```

3. **Run the discovery**

   For details running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Method 1:

   - Run the **SAP TCP Ports** job to discover SAP ports.
   - Run the **SAP ABAP Connection by SAP JCO** job.
   - Run the **SAP Solution Manager Topology by SAP JCO** job to discover complete topology of registeredSAP systems.
   - Run the **SAP Solution Manager by SAP JCO** job to discover the SAP business layer .

   Method 2:

   - Run the **Host Resources by ...** jobs to discover SAP (ABAP or J2EE) Application Server and/or SAP (ABAP or J2EE) Central Services.
   - Run the **SAP System by Shell** job to create a SAP system CI (but without defining whether it is the SAP Solution Manager).
   - Run the **SAP ABAP Connection by SAP JCO** job.
   - Run the **SAP Solution Manager Topology by SAP JCO** job to discover complete topology of

registererd SAP systems.

- Run the **SAP Solution Manager by SAP JCO** job to discover the SAP business layer .

During the run of the **SAP ABAP Connection by SAP JCO** job, the SAP Systems that are defined as the SAP Solution Manager are triggered on these two jobs: **SAP Solution Manager Topology by SAP JCO** and **SAP Solution Manager by SAP JCO** job.

Troubleshooting and Limitations

Problem. The SAP discovery fails and a Java message is displayed:

```
This application has failed to start because MSVCR71.dll was not found.
```

Solution. Two .dll files are missing. For the solution, read Note #684106 in https://websmp205.sap-ag.de/~form/sapnet?_FRAME=CONTAINER&_OBJECT=012003146900000245872003.

Chapter 48: Siebel Discovery

This chapter includes:

Overview

Using the Siebel adapters, you can run an automatic Siebel discovery to create the Siebel world, together with its components, inside HP Universal CMDB. During discovery:

- All Siebel-related IT entities that reside in the organization are discovered, and configuration items (CIs) are written to the CMDB.
- The relationships between the elements are created and saved in the CMDB.
- The newly generated CIs are displayed when the Siebel Enterprises view is selected in View Explorer under the Siebel Enterprises root CI.

Note: Verify that all Siebel server IP addresses are included in the range. If not all servers can be covered with one IP range, you can split the range into several ranges.

Supported Versions

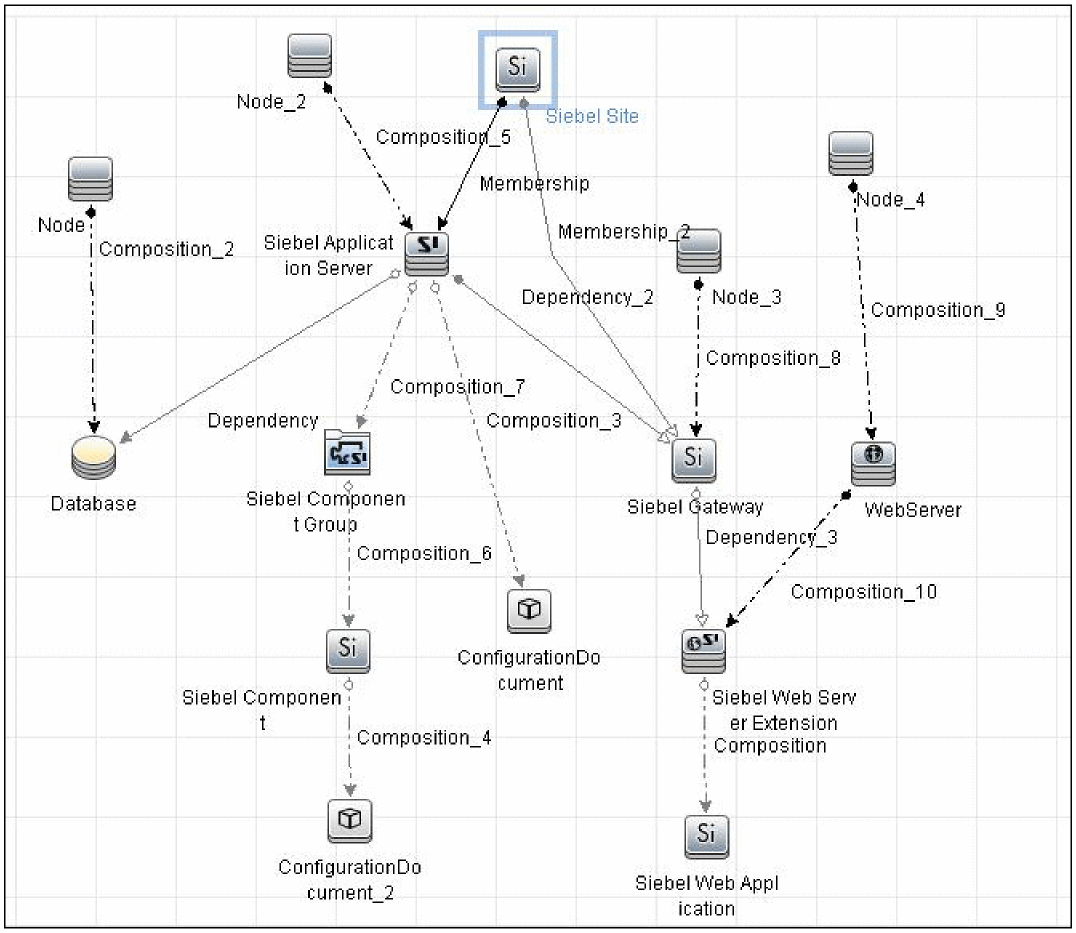
This discovery solution supports the following servers:

- Siebel 7.5
- Siebel 7.7
- Siebel 8.0
- Siebel 8.1

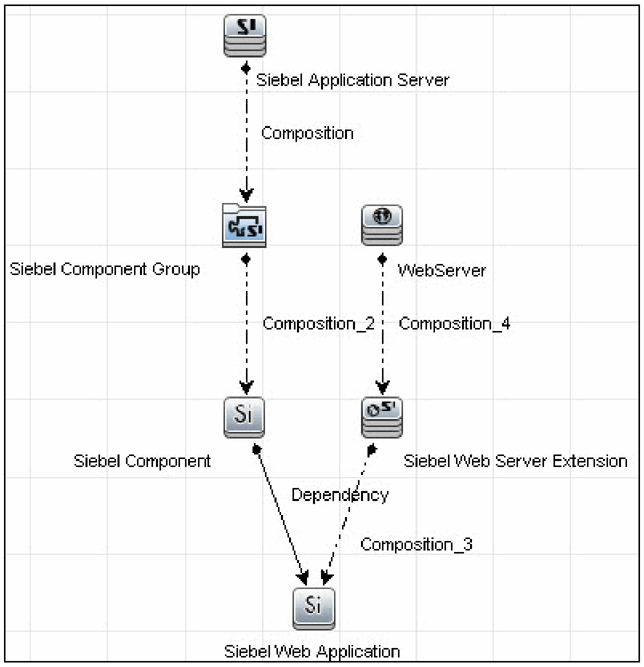
Topology

The following images display the Siebel topologies:

Siebel Topology View



Siebel Web Topology View



How to Discover Siebel Topology

This task describes how to discover Siebel topology. It includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisites - Other" below
- "Run the discovery" on the next page

Prerequisite - Set up protocol credentials

Set up the following protocols:

Platform	Protocol
Windows	• WMI protocol • NTCMD protocol • Siebel Gateway protocol
UNIX	• SSH protocol • Telnet protocol • Siebel Gateway protocol

Note: The Siebel Gateway protocol allows the user to specify which port is used during connection to the gateway.

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

Prerequisites - Other

The driver tool is used to extract data about the enterprise structure from Siebel.

Note:

- If you are working with different versions of Siebel in your organization, make sure you use a driver tool with a version that is appropriate for the Siebel server.

- If the Data Flow Probe is installed on a 64-bit machine on a Windows platform, place the **ntdll.dll**, **MSVCR70.DLL**, and **msvcp70.dll** drivers together with the Siebel drivers in the Siebel driver folder on the Probe machine. You enter details of this folder in the Siebel set of credentials (**Path to Siebel Client**). These drivers usually exist on a 32-bit machine and can be copied to the 64-bit machine.

  For details, see "Siebel Gateway Protocol" in the *HP Universal CMDB Data Flow Management Guide*.

To copy the driver tool to the Data Flow Probe:

1. Copy the driver Command Line Interface (CLI) tool from the Siebel server to any folder on the Data Flow Probe machine.

2. (Recommended) Run the Siebel connection test to validate the driver installation. To run the connection test, open the command line on the Data Flow Probe machine and change directory to the location of the **driver.exe** file.

3. Run from the command line:

   ```
   >driver /e [site_name] /g [gateway_host] /u [username] /p [password]
   ```

   If the connection is established successfully, the Command Prompt window displays the `driver` prompt and a status message about the number of connected servers.

Run the discovery

1. To trigger the discovery of Siebel networking features, add a Network CI to the CMDB. For details, see "New CI/New Related CI Dialog Box" in the *HP Universal CMDB Modeling Guide*.

2. In the Universal Discovery window, activate jobs in the following order:

   a. Range IPs by ICMP, or Range IPs by nmap

   b. Host Connection by Shell

   c. Host Connection by WMI

   d. Host Resources by Shell

   e. Host Resources by WMI

f. Host Applications by Shell

g. Host Applications by WMI

h. Siebel Web Applications by NTCMD or UDA, and/or Siebel Web Applications by TTY

i. Siebel Gateway Connection

j. Siebel Application Servers

k. Siebel Application Server Configuration

l. Siebel DB by NTCMD or UDA, and/or Siebel DB by TTY

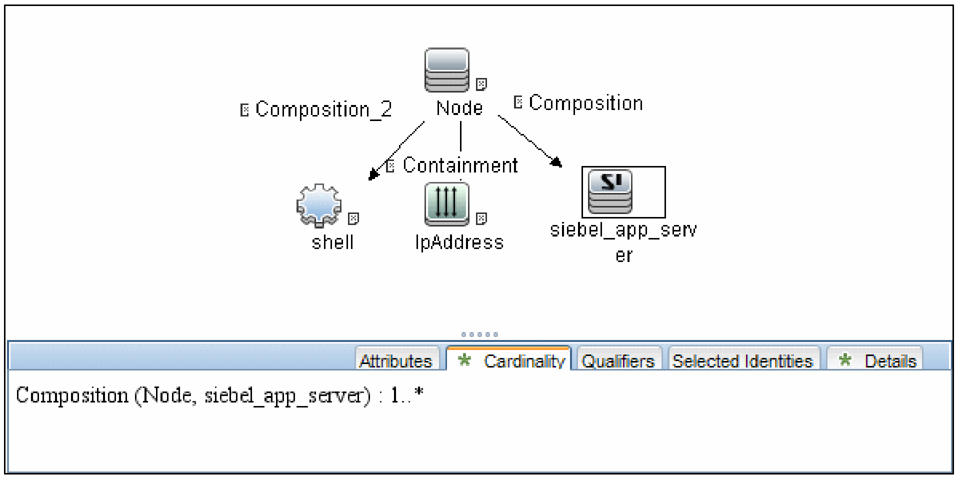
Note: The following enrichment adapters automatically run in the background during discovery:
Siebel_Route_WebApp_To_Component. Builds the route between Siebel Web Application CIs and Siebel Component CIs.
Siebel_Web_To_Middle_Tier. Builds the route between the Web tier and the middle tier when the Siebel enterprise uses a Resonate server for load balancing.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Siebel Application Server Configuration Job

This section includes details about the job.

Trigger Query



Composition (Node, siebel_app_server) : 1..*

Adapter

This job uses the **SIEBEL_DIS_APP_SERVER_CONFIG** adapter.

Used Scripts

- file_ver_lib.py
- siebel_discover_appserver_config.py

Discovered CITs

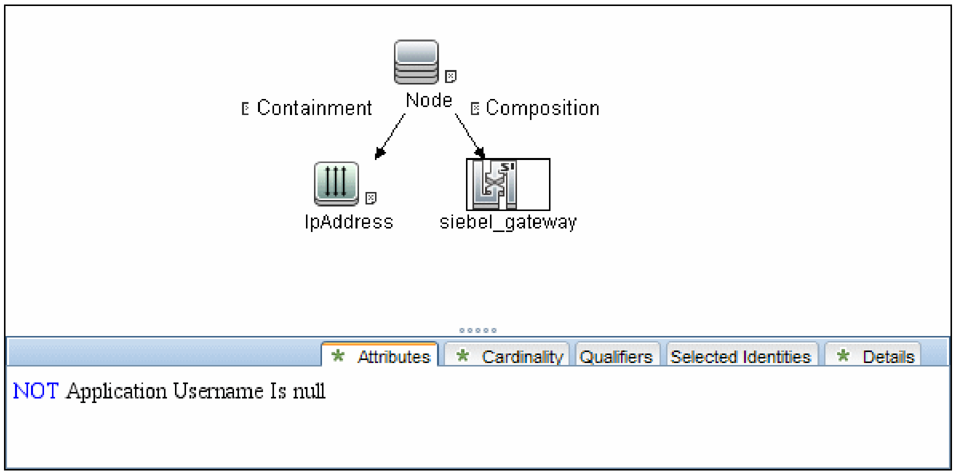
- Composition
- ConfigurationDocument
- Siebel Application Server

Note: To view the topology, see "Siebel Topology View" on page 642.

Siebel Application Servers Job

This section includes details about the job.

Trigger Query



NOT Application Username Is null

Adapter

This job uses the **SIEBEL_DIS_APP_SERVERS** adapter.

Used Scripts

- siebel_common.py
- siebel_discover_enterprise.py

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Dependency
- IpAddress
- Membership
- Node

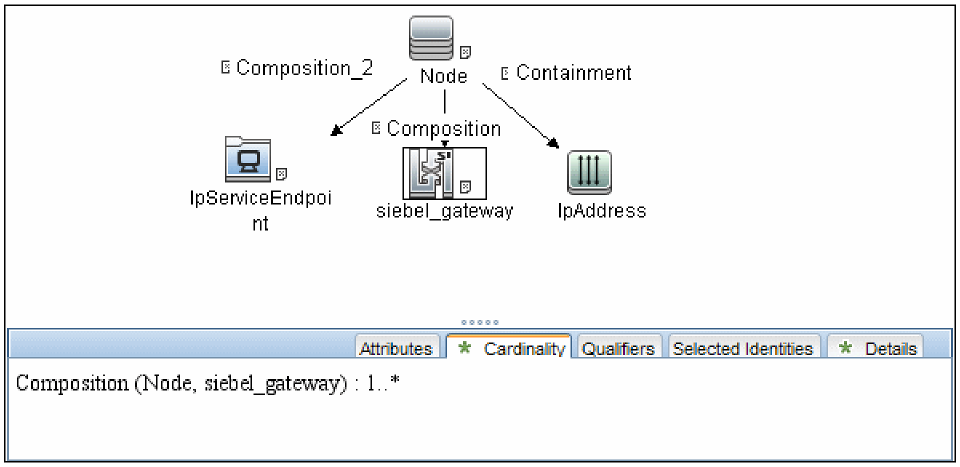
- Siebel Application
- Siebel Appication Server
- Siebel Component
- Siebel Component Group

Note: To view the topology, see "Siebel Topology View" on page 642.

Siebel Gateway Connection Job

This section includes details about the job.

Trigger Query



Composition (Node, siebel_gateway) : 1..*

Adapter

This job uses the **SIEBEL_DIS_GATEWAY_CONNECTION_(GTWY)** adapter.

Used Scripts

- siebel_common.py
- siebel_discover_gateway.py

Discovered CITs

For details on the CIs that are discovered, see the Statistics table in the **Details** tab.

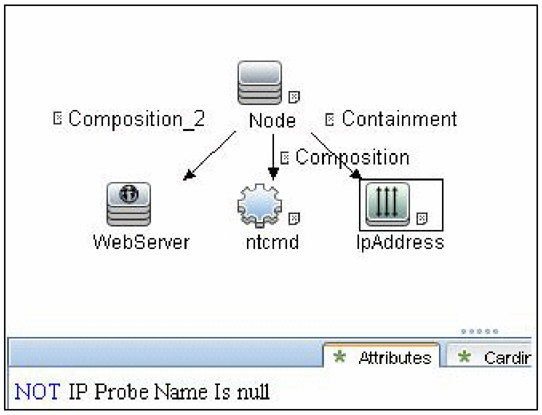
- Composition
- Membership
- Siebel Enterprise
- Siebel Gateway

Note: To view the topology, see "Siebel Topology View" on page 642.

Siebel Web Applications by NTCMD or UDA Job

This section includes details about the job.

Trigger Query



NOT IP Probe Name Is null

Adapter

This job uses the **SIEBEL_DIS_WEBAPPS_NT** adapter.

Used Scripts

- NTCMD_HR_REG_Software_Lib.py

- siebel_discover_wse.py

Discovered CITs

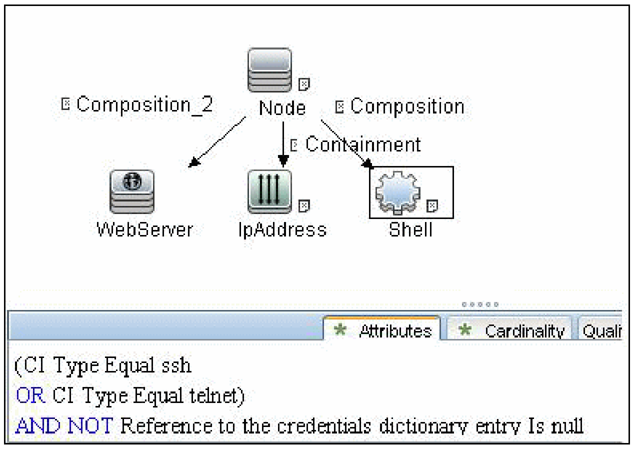
- Composition
- Configuration Document
- Containment
- Dependency
- IpAddress
- Node
- Route
- Siebel Enterprise
- Siebel Gateway
- Siebel Web Application
- Siebel Web Server Extension
- WebServer

Note: To view the topology, see "Siebel Web Topology View" on page 643.

Siebel Web Applications by TTY Job

This section includes details about the job.

Trigger Query



(CI Type Equal ssh
OR CI Type Equal telnet)
AND NOT Reference to the credentials dictionary entry Is null

Adapter

This job uses the **SIEBEL_DIS_WEBAPPS_UNIX** adapter.

Used Scripts

- siebel_discover_wse.py
- NTCMD_HR_REG_Software_Lib.py

Discovered CITs

- Composition
- Configuration Document
- Containment
- Dependency
- IpAddress
- Node
- Route
- Siebel Enterprise

- Siebel Gateway
- Siebel Web Application
- Siebel Web Server Extension
- WebServer

Note: To view the topology, see "Siebel Web Topology View" on page 643.

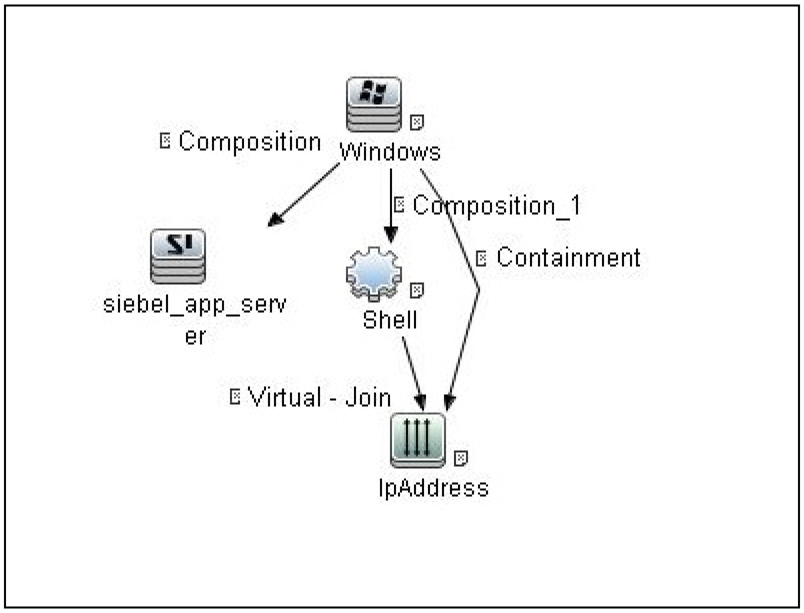
Parameters

- **eappsCfgPath.** The path to the Siebel Webserver Extension configuration file (eapps.cfg).

Siebel DB by NTCMD or UDA Job

This section includes details about the job.

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_DB_NT** adapter.

Used Script

- siebel_discover_odbc.py

Discovered CITs

- Composition
- Containment
- Database

- Dependency
- IpAddress
- Node

Note: To view the topology, see "Siebel Topology View" on page 642.

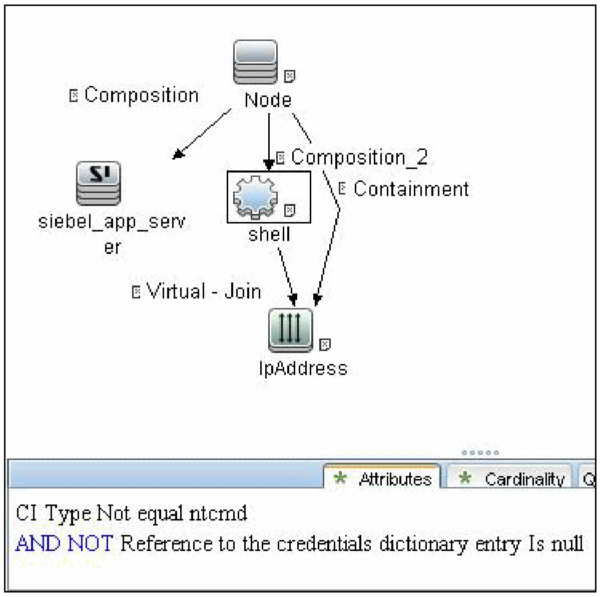
Parameter

- **oracle_name.** Can include several ORACLE_NAME paths (for different machines), comma separated. If empty, a hard-coded (in the script) registry is used.

Siebel DB by TTY Job

This section includes details about the job.

Trigger Query



CI Type Not equal ntcmd
AND NOT Reference to the credentials dictionary entry Is null

Adapter

This job uses the **SIEBEL_DIS_DB_UNIX** adapter.

Used Script

- siebel_discover_odbc.py

Discovered CITs

- Composition
- Containment
- Database
- Dependency
- IpAddress
- Node

Note: To view the topology, see "Siebel Topology View" on page 642.

Troubleshooting and Limitations

The Siebel DB by TTY job cannot discover virtual Siebel application servers (with a different name and configuration to the actual Siebel application server) running on UNIX machines.

Chapter 49: Cisco UCS

This chapter includes:

Overview

Cisco UCS manages hardware and software in datacenters. This integration solution is based on its XML API to discover managed topologies. This solution contains 3 jobs.

Cisco UCS Connection Job

The Job is used to discover Cisco UCS as running software which will be trigger of the job 'Cisco UCS Topology'.

Versions

The package supports Cisco UCS version 2.2(1b).

Prerequisites

The job needs UCS credential which include username and password.

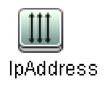
Adapter Information

This job uses the **Cisco_UCS_Connection** adapter.

Input CIT

ip_address

Trigger TQL



Used Scripts

- ucs_connection_main.py
- ucs_connection_data_manager.py
- ucs_client.py
- ucs_base.py
- ucs_decorators.py

Discovered CITs

- running _software

Global Configuration File

- cisco_ucs/ucs_mapping.xml

Workflow

1. Get credentials for the trigger IP.
2. Iterate credentials, create UCS client to login. If successfully connected, report a UCS running software with the connected credential.

Cisco UCS Topology Job

The Job is used to discover the Cisco UCS topology with UCS running as an input trigger.

Versions

The package supports Cisco UCS version 2.2(1b).

Prerequisites

The job needs UCS credential which include username and password.

Adapter Information

This job uses the **Cisco_UCS_Connection** adapter.

Input CIT

ip_address

Trigger TQL

Used Scripts

- ucs_topology_main.py
- ucs_pull_base.py
- ucs_connection_data_manager.py
- ucs_client.py
- ucs_base.py
- ucs_mapping_file_manager.py
- ucs_mapping_implementation.py
- ucs_mapping_interfaces.py
- ucs_validators.py
- ucs_decorators.py

Discovered CITs

- chassis
- fchba
- fcport
- fcswitch
- hardware_board
- node
- physical_port
- rack

Global Configuration File

- cisco_ucs/ucs_mapping.xml

Workflow

1. Get credential from UCS running software.
2. Connect UCS system by the credential and fetch data from UCS by its XML API.
3. Parse the data from UCS and map them to UCMDB, then report CIs to UCMDB.

Cisco UCS Manual Job

The Job is used to discover Cisco UCS topology by a configuration file containing lines of URLs.

Versions

The package supports Cisco UCS version 2.2(1b).

Prerequisites

The job needs UCS credential which include username and password.

Adapter Information

This job uses the **Cisco_UCS_Topology** adapter.

Input CIT

discoveryprobegateway

Trigger TQL

Used Scripts

- ucs_manual_main.py
- ucs_pull_base.py
- ucs_connection_data_manager.py
- ucs_client.py
- ucs_base.py
- ucs_mapping_file_manager.py
- ucs_mapping_implementation.py
- ucs_mapping_interfaces.py
- ucs_validators.py
- ucs_decorators.py

Discovered CITs

- chassis
- fchba
- fcport
- fcswitch
- hardware_board
- node
- physical_port
- rack

Global Configuration File

- cisco_ucs/ucs_mapping.xml

Workflow

1. Read ucs_url_list.conf and parse it to a list of URL.
2. Pick one URL and try all UCS credentials one by one until connected, then fetch data from UCS by its XML API
3. Parse the data from UCS and map them to UCMDB, then report CIs to UCMDB
4. Iterate steps 2-3 over the rest of URLs

Troubleshooting

Issue: You receive the following error message in the Communication log: URLError: <urlopen error (-1, 'SSL handshake exception

Solution 1: Https server doesn't own a valid SSL certificate (E.g. self-signed certificate or expired), for this case, make sure enable the 'Trust All SSL Certificates' options in current used UCS credential.

Solution 2: The server is not an Https server but you enabled the SSL in the credential. Disable the 'Use HTTPS for connection' option for the credential.

Part 8: Hosts and Resources

Chapter 50: Application Signatures

This chapter includes:

Overview

Application signatures is a method of identifying applications running on a target host, based on previously discovered host resources data; specifically:

- Processes
- Open ports (optional)

However, depending on context, application signatures can mean the:

- Method of identification in general
- Discovery module that implements the method
- Configuration file describing signatures

Method Capabilities

The capabilities of the application signatures method include:

- Identification of applications based on full or partial match of process names.
- Identification of applications based on presence of specific open ports.
- Identification of applications based on presence of specific substrings in command lines of the processes.
- Distinction between instances of applications where several are running on the same host. (Limitations apply.)
- Reporting specific subclasses of RunningSoftware CIT if enough identification information is available.
- Completing additional attributes for RunningSoftware CIs. Data can be taken from processes by using parse rules.
- Support for plug-ins. Whenever an application is found, it is possible to execute additional commands and enrich the topology of the application with more data.

For more information, see "Application Signatures Mechanism" on page 672.

Signature Repository

Application Signatures uses a repository of signatures stored in a configuration file called **applicationsSignature.xml**. This XML file contains a number of Application-Component elements, each describing a signature for specific application or flavor of application. For example, the signature for Microsoft IIS Web server may look as follows:

Microsoft IIS Web server signature

```
<Application-Component name="Microsoft IIS WebServer" ci_type="iis"
category="Web Server"
     vendor="microsoft_corp" installedSoftwareName="\s*IIS\s+.*" supported_
versions="5.1, 6.0, 7.0">
          <process name="inetinfo.exe" ports="all,None" cmdline=""
          description="Microsoft Internet Information Service process." />
          <process name="w3wp.exe" ports="None" cmdline=""
          description="Microsoft Internet Information Service worker process."
/>
</Application-Component>
```

For more information, see "Application Signatures Configuration File" on page 679.

Input Data

Application Signatures uses the following input data:

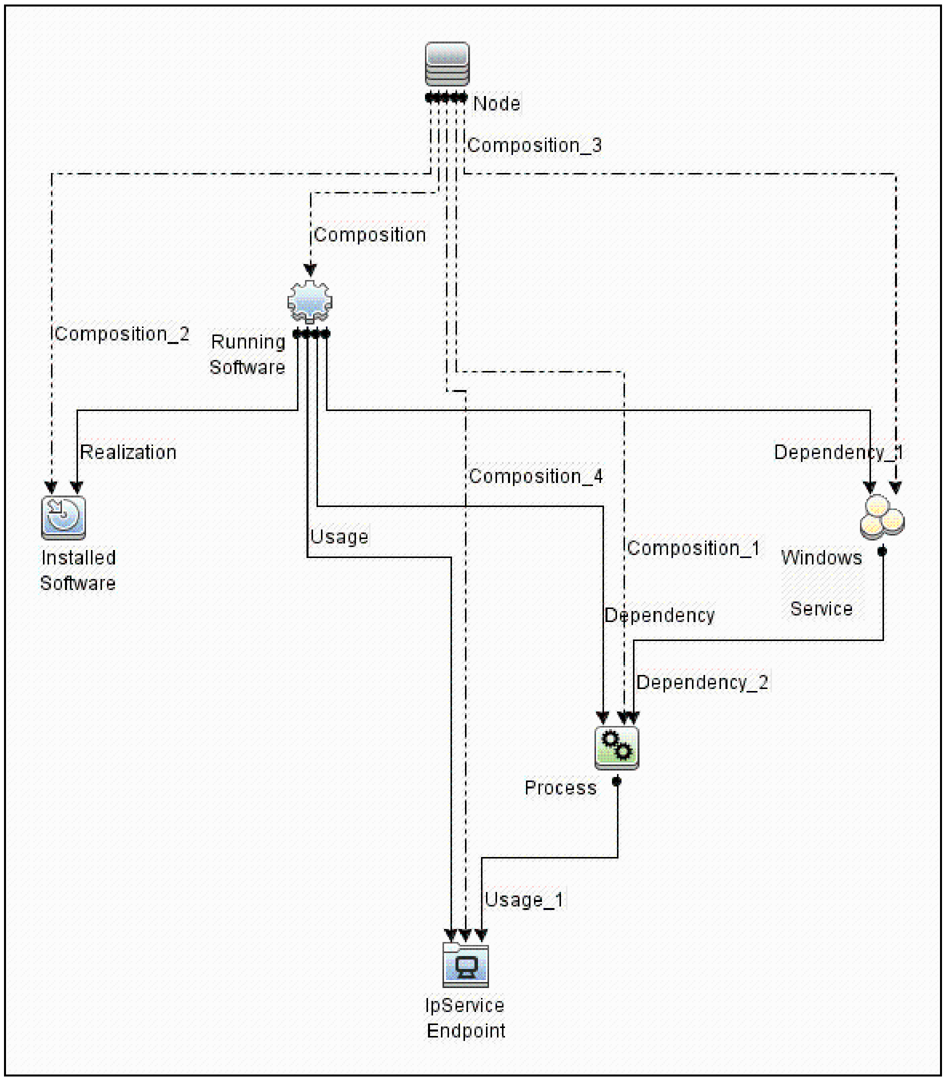
- Processes running on the host, including"
  - name
  - PID
  - command line, executable path, arguments
  - owner
- Ports of processes (optional)
  - regular open ports and listening ports
  - which interfaces the ports are open on

- Installed Software (optional)
- Services (optional)

Processes and ports information is used in identification of applications. Information about services and installed software is not used in identification, but allows the building of relationships to corresponding topologies.

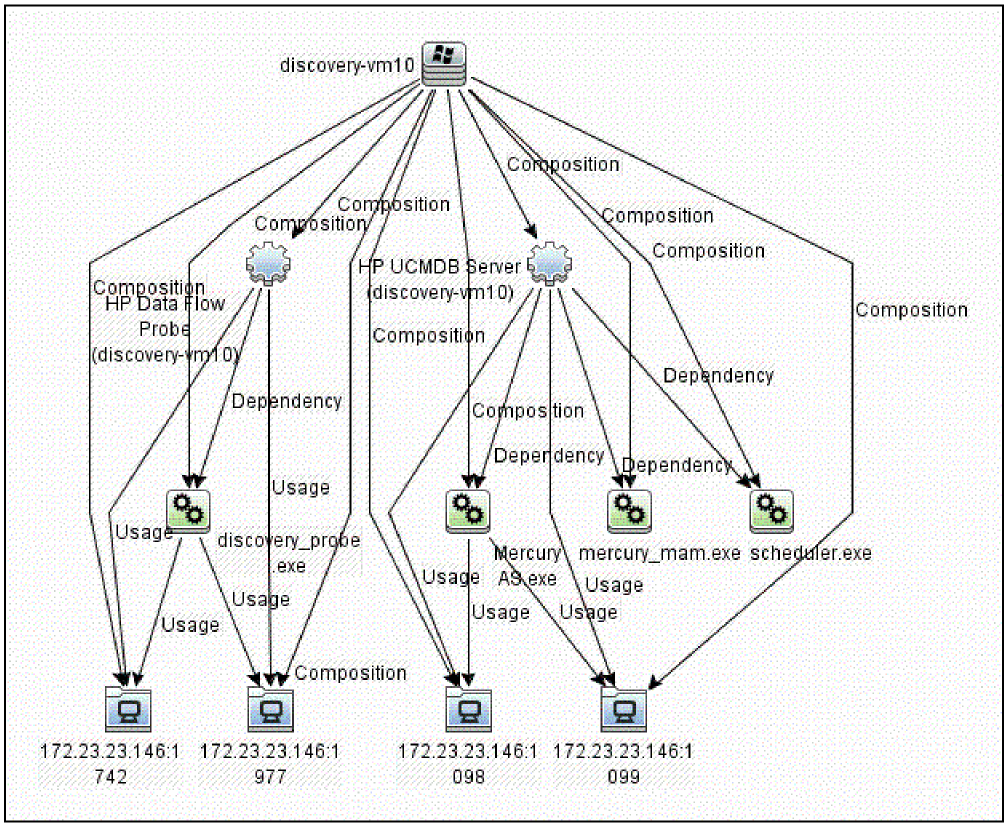
Topology

The main element of Application Signatures topology is the RunningSoftware CIT, representing running applications. The following image shows the type of topology reported by Application Signatures.



Topology Example

The following image is an example of topology found which includes HP UCMDB Server and Data Flow Probe running on the same host.



Application Signatures Mechanism

This part includes:

Jobs and Adapters

Application Signatures run as a part of host resources jobs. There are several adapters implementing host resources discovery, and for each there are two jobs:

- **Adapter: Host Resources by TTY (TTY_HR_All.xml)**
  - Job: Host Applications by Shell
  - Job: Host Resources by Shell
- **Adapter: Host Resources by WMI (WMI_HR_ALL.xml)**
  - Job: Host Applications by WMI
  - Job: Host Resources by WMI
- **Adapter: Host Resources by SNMP (SNMP_HR_ALL.xml)**
  - Job: Host Applications by SNMP
  - Job: Host Resources by SNMP
- **Adapter: Host Resources by PowerShell (PowerShell_HR_All.xml)**
  - Job: Host Applications by PowerShell
  - Job: Host Resources by PowerShell

Jobs and Default Behavior

The jobs **Host Resources by X**, by default:

- Discover and report CPUs
- Discover and report Disks
- Discover and report Memory
- Discover and report Users
- Discover and report Network Shares
- Discover processes, and save data to Probe database
- Discover process ports, and save data to Probe database
- Discover and report process-to-process topology
- Do not run Application Signatures discovery

The jobs **Host Applications by X**, by default:

- Discover processes, and save data to Probe database
- Discover process ports, and save data to Probe database
- Discover installed software
- Discover and report process-to-process topology
- Run Application Signatures discovery

Note: None of the listed jobs discover Services by default.

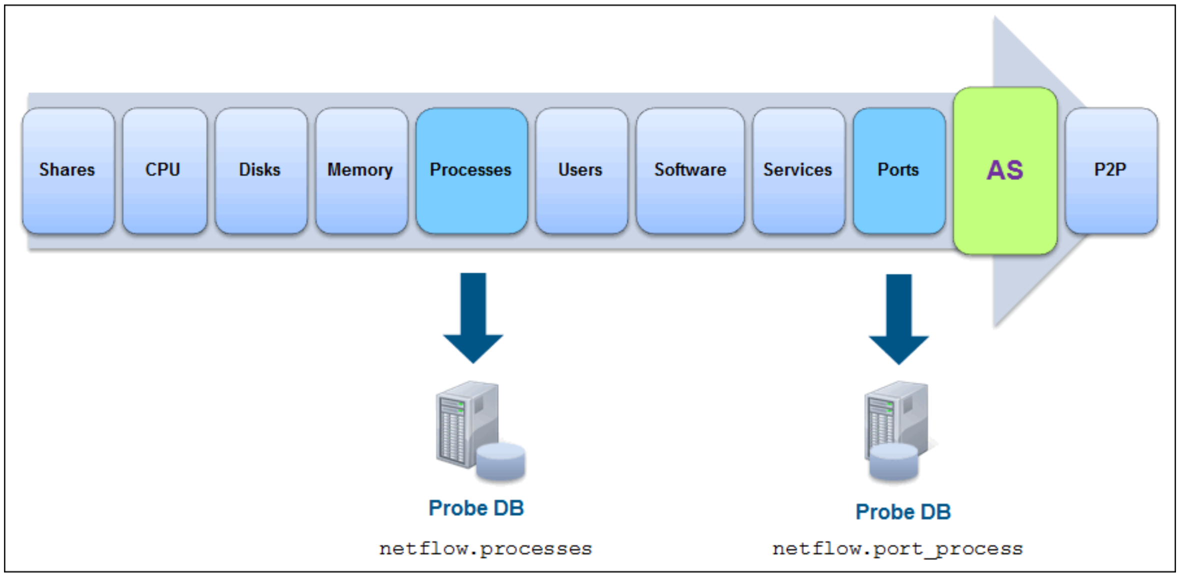
Jobs provide parameters that control which part of the host resources topology should be reported. You may create a new custom job based on the same adapters, where the parameters can be changed to desired values. See "Host Resources and Applications Discovery" on page 698.

Application Signatures discovery is also available as a part of Management Zones discovery. To invoke applications discovery using Application Signatures, you should configure a Basic Software Configuration

Discovery. See "Universal Discovery Activities" in the *HP UCMDB Universal Discovery Content Guide - Discovery Activities* document.

Host Resources Jobs Flow

The following image shows the general flow of Host Resources jobs, though some parts of the flow may be enabled or disabled via parameters:



The flow of host resources jobs related to Application Signatures is as follows:

1. Processes discovery runs as a part of the Host Resources job. Discovered processes are saved to the Probe's database (in the table netflow.processes). If the job's parameter **discoverProcesses** is set to true, these processes are added to the results vector at this point. Otherwise the processes are not reported immediately.

2. TCP discovery runs, which discovers all open ports; both listening ports and regular client ports. This data is saved to the Probe's database (in the table table netflow.port_process).

3. The Application Signatures engine is configured, and discovered processes and open ports are passed to the engine as input data. If Services and Installed Software objects were discovered, they are also passed to the Application Signatures engine.

4. Application Signatures runs, and performs identification of applications, and reports corresponding topology.

5. Process-to-process discovery runs, reporting client-server links between processes.

Application Signatures Flow

Application Signatures performs the following activities to identify and reports applications and related topology:

1. Engine reads configuration from applicationsSignature.xml file.
2. Engine tries to relate discovered processes to each signature.
3. Engine validates and discards unmatched signatures and signatures with unsatisfied requirements.
4. Engine makes a decision about the number of instances of applications for each signature.
5. Engine creates OSH objects corresponding to applications.
6. Engine performs updates of application attributes by evaluating expressions using parse rules if they are present. Attribute updates may be optional or required. If a parse rule used in the expression of required attribute fails to match, the application instance is skipped. Optional attribute update has no such effect.
7. Engine finds all suitable plug-ins for particular application and executes them. Plug-in has ability to run additional commands with the same client, update attributes, enhance topology, and even make a decision to skip the application.
8. Engine forms results vector with all application objects. Also at this stage, it links application and processes with installed software and services.

Identifying application instances

Many applications support instancing - multiple independent sub-applications running on the same host. It is common for such instances to have names or other identifiers in order to distinguish them. Instancing support in Application Signatures is based on the idea that for each instance, there is a corresponding designated process running. It means there is a 1:1 relationship between the number of such processes and the number of application instances. Identifying and marking such processes in the Application Signatures configuration file enables reporting of such instanced applications. For this purpose, the **main-process** boolean attribute was introduced for the element **process** in the signatures.

The attribute **main-process** affects the topology of the application as follows:

- When the **main-process** attribute is not set for any process in **application-component**, or is set to **false** (default), a single **RunningSoftware** CI is created, and all processes are linked to this CI.
- When the **main-process** attribute is set to **true** for a process, the process is regarded as the one that identifies the application instance. The number of such processes defines the number of application instances and the number of **RunningSoftware** CIs created. Each instance of main process is linked to only one corresponding **RunningSoftware** CI. All other processes are considered 'shared' and are linked to all **RunningSoftware** CIs created.

Example of signature of instanced application

```
<Application-Component name="A" >
     <process name="P1" main-process="true" ports="None" cmdline=""
      required="true" />
     <process name="P2" ports="None" cmdline="" />
</Application-Component>
```

- Signature of application A consists of process **P1** with **main-process** attribute set to **true** and process **P2**.
- Discovered data contains 2 instances of process **P1** (P11, P12) and 2 instances of process **P2** (P21, P22).
- Resulting topology is the following:
- Two **RunningSoftware** CIs **A1** and **A2** since there are two processes marked as main.
- **A1** is linked to processes **P11, P21,** and **P22**.
- **A2** is linked to processes **P12, P21,** and **P22**.

Note: Instancing for applications without designated process(es) per instance with a 1:1 relationship, is not supported.

Attribute updates and expressions evaluation

Application Signatures supports attribute updates - declarations in the signature of applications that specific attributes of reported **RunningSoftware** CIs should be filled with specified values. Values may be:

- constants
- expressions containing strings and references to values of evaluated parse rules

A parse rule is a pattern declared in the signature that is matched against the data of the applications, such as command lines or paths of the processes. Patterns of a parse rule either match or do not match. A parse rule pattern is a regex (regular expressions) pattern. When a parse rule matches, corresponding matched string or groups can be used in expressions of attribute updates. If a parse rule is declared to match against the property of process, matching of the parse rule is performed against every process, one by one until the first match.

The following is an example of signature for the **Active Directory Application Mode** application, where attribute update with a parse rule is used:

Active Directory Application Mode with attribute updates

```
<Application-Component name="Active Directory Application Mode" ci_type="adam"
category="Enterprise App" vendor="microsoft_corp">
     <parse-rule id="instance_name" name="cmdline">.+\s+-sn:([\w-]
       +).*</parse-rule>
     <process name="dsamain.exe" main-process="true" ports="ldap,636,
       None" required="true" cmdline="" description="Main AD
       Application Mode process" />
<attribute name="name" value="${instance_name(1)}" type="string" />
</Application-Component>
```

```
<Application-Component name="Active Directory Application Mode" ci_type="adam"
category="Enterprise App" vendor="microsoft_corp">
     <parse-rule id="instance_name" name="cmdline">.+\s+-sn:([\w-]
       +).*</parse-rule>
     <process name="dsamain.exe" main-process="true" ports="ldap,636,
       None" required="true" cmdline="" description="Main AD
       Application Mode process" />
<attribute name="name" value="${instance_name(1)}" type="string" />
</Application-Component>
```

The flow for this signature is as follows:

- Process **dsamain.exe** that is found on the target host allows this signature to match and produces **RunningSoftware** of type **adam.**
- When the engine gets to the point of attribute updates, all parse rules are matched for this application:

  - The pattern of parse rule **instance_name** is matched against command lines of processes found.
  - The pattern uses capturing groups; the matched pattern exposes both the whole matched string and group 1 captured. This group contains the value of ADAM instance.
- Expression of attribute update ${instance_name(1)} is evaluated, and it translates to a string containing the instance name of ADAM extracted from the command line,
- The value is reported to attribute **name** of application CI.

Ports matching and reporting

The element **process** in the signature can specify port values that affect whether this **process** element matches, and what ports are reported. Ports are specified using the attribute **ports**: a comma-separated list of values.

The following rules apply:

- If among the values of the **ports** attribute there is a 'None' string, the **process** element always matches, regardless of ports discovered for this process.
- Otherwise, in order for the element **process** to match, the discovered process should have all corresponding ports open that are specified in the attribute.
- Only ports specified in the attribute **ports** are reported
- Application Signatures can also report all ports of the application listened to. There are two ways to enabled this:
  - Only for specific application and specific process by including the keyword 'all' in the value of the **ports** attribute
  - Globally via boolean parameter **discoverAllListenPorts** in the configuration file **globalSettings.xml**. For example:

    globalSettings.xml

    ```
    <property name="discoverAllListenPorts">false</property>
    ```

Application Signatures Configuration File

The configuration file **applicationsSignature.xml** is a repository of signatures for application signature discovery. Each signature contains a set of processes associated with the application. An example signature follows:

Signature for PostgreSQL Database Application

```
<Application-Component name="PostgreSQL" category="Database" vendor="PostgreSQL"
  supported_versions="7.x, 8.x" installedSoftwareName="PostgreSQL.*">
  <process name="pg_ctl.exe" ports="postgresql,None" cmdline=""
description="PostgreSQL Server." />
  <process name="postmaster" ports="postgresql,None" cmdline=""
description="PostgreSQL Server." />
</Application-Component>
```

Configuration File Structure

The element **Applications** is the root of the configuration file, it contains a list of signatures represented as **Application-Component** elements. The additional element **Default-Application-Type** specifies the CIT to use when reporting applications.

Signature for PostgreSQL Database application

```
<Applications parserClassName="com.hp.ucmdb.discovery.library.
communication.downloader.cfgfiles.ApplicationSignatureConfigFile">

     <Default-Application-Type>running_software</Default-Application-Type>

     <Application-Component name= ... >
         <process name= ... />
     </Application-Component>

     ...

     <Application-Component name= ... >
          <process name= ... />
     </Application-Component>

</Applications>
```

Elements and Attributes

Application-Component element

The **Application-Component** element defines a signature for a specific application, or type of application. It may contain the attributes described in the following table:

Attribute	Optional?	Description
name	No	The name of application. This value is reported to the **DiscoveredProductName** attribute of the application CI.
app_id	Yes	The ID of the signature. If this attribute is not set, the value of the attribute **name** is used for ID. **Note:** All ID values must be unique within the configuration file.
category	Yes	The category of the application. This value is passed to the application's **application_category** attribute. Examples: Database, Cluster.
vendor	Yes	The application vendor. This value is passed to the application's **vendor** attribute. Example: hewlett_packard_co.
supported_versions	Yes	A list of versions for which the signature applies. This information has no impact on reported topology and is present for informational purposes only.
installedSoftwareName	Yes	The regex (regular expression) pattern used to report relationships with Installed Software. Whenever the name of Installed Software matches the pattern, corresponding relationships are reported.

Attribute	Optional?	Description
ci_type	Yes	The CIT name that should be used when the application is reported. If not specified, the default CIT is used: RunningSoftware. Example: <Application-Component name="Oracle DB" ci_type="oracle"> says that this application should be reported as CIT "oracle".
discover	Yes	A boolean attribute which enables (**true**) or disables (**false**) the signature. **Default:** True.

The **Application-Component** should contain one or more **process** elements, which define what processes should be discovered in order for this application to be reported. A process element may contain the attributes described in the following table:

Attribute	Optional?	Description
name	No	The name of the process.
ports	No	A comma-separated list of ports the process should listen to, or should have open, in order to be matched. Each port may have one of the following values: • None. The process matches even if it does not listen to any port. • Numeric value or a named port that resolves to numeric value. The process should listen to this port, or should have this port open in order to match. Examples: 1521, mysql. Port names are resolved against the portNumberToPortName.xml configuration file. • All. Specifies that all ports listened to should be reported All ports matched through this pattern are reported as IpServiceEndpoint CIs.
cmdline	Yes	The substring that should be present in the command line of the process for it to match.
description	Yes	The description of the process. The value of this attribute is passed to the **description** attribute of processes CIs that match this process element.
startswith	Yes	A boolean attribute that enables partial matching of the process names. When this attribute is set to **true**, any process whose name starts with the string specified in the attribute **name**, matches.

Attribute	Optional?	Description
required	Yes	A boolean attribute which, if set to **true**, specifies that the process is required for the application to be reported. If this process element does not match to any process, the application is not reported.
main-process	Yes	A boolean attribute indicating, if set to **true**, that this process is the main process of the application. For each main process found on the host, a separate application topology is reported.
ignore-case	Yes	A boolean attribute specifying whether the process name matching is case-sensitive (**true**) or not (**false**). **Default:** False.

Parse-rule element

A parse-rule element defines a parse rule that is applied to a specific property of a process that is found, in order to extract some values. A parse rule is essentially a regular expression that should match against a process property. If this regex contains capturing groups, it is possible to extract the values of these groups and use them in expressions of attribute updates. This regular expression is written in the text part of the element. A parse-rule element may contain the attributes described in the following table:

Attribute	Optional?	Description
id	No	ID of the parse rule; a unique string identifying this parse rule.
name	No	Name of process property this rule should be matched against. Supported properties: • **cmdline** • **ip** • **name** • **owner** • **path** • **port**

Attribute	Optional?	Description
method	Yes	The method to use when matching the pattern, being one of the following: • **match**. The entire property is matched against the pattern. It behaves like the regular expression 'match' operation. • **search**. The pattern is searched for within the string. It behaves like the regular expression 'search' operation. **Default:** Match.

Attribute element

When the element **Attribute** appears under an **Application-Component** element, it means that the specified attribute of the reported application CI should be set to the specified value. The value may be a constant, string, or an expression with parse rule references. An attribute element may contain the attributes described in the following table:

Attribute	Optional?	Description
name	No	The name of the attribute to set.
value	No	An expression that defines the value of the attribute. This expression may contain regular text which will be used without modification, and references to parse rules. Parse rule reference is defined as follows: • ${rule_id} - A whole match of the parse rule with ID rule_id is inserted into the resulting string during expression evaluation. • ${rule_id(group_numer)} - When parse rule with ID rule_id matches the value of capturing group number group_number is inserted into result string.
required	Yes	A boolean attribute specifying whether (**true**) or not (**false**) the attribute is required. If the attribute is required, an application is skipped whenever the expression of attribute fails evaluation. If the attribute is not required and expression evaluation fails, the application is reported without this attribute set. **Default:** True.

Clustered-application element

An Application-Component element may contain the additional element **clustered-application** which marks the application as clustered. Marking an application as clustered results in reporting the parent Node CI for this particular application as a weak Node by the application IP. Applications not marked as clustered use the parent Node CI restored from CMDB ID.(The default behavior).

Example of usage of clustered-application element

```
<Application-Component name="MSSQL DB" category="Database" ci_type="sqlserver"
vendor="microsoft_corp">

     <clustered-application/>

</Application-Component>
```

Application-ip-source element

A process element may contain the additional element **application-ip-source**. This element marks processes as a source of application IP, and affects how an application IP is chosen for a particular application. When one of the processes is marked as a source of application IP, only IP addresses this process listens to are considered for application IP. When none of the processes are marked, all IPs of all processes are considered for application IP. (The default behavior).

Example of usage of application-ip-source element

```
<Application-Component name="MSSQL DB" category="Database" ci_type="sqlserver"
vendor="microsoft_corp">

     <process name="sqlservr.exe" main-process="true" ports="sql,None"
cmdline="" required="true" description="Provides storage, processing and
controlled access of data and rapid transaction processing." >

          <application-ip-source />

     </process>

</Application-Component>
```

Service-endpoint-name element

A process element may contain an additional element: **service-endpoint-name**. When a process has this additional element, all IpServiceEndpoint CIs of this process are reported with the specified name. If this element is not used, the name of IpServiceEndpoint CIs is resolved using the portNumberToPortName.xml file. (This is the default behavior.)

Example of usage of service-endpoint-name element

```
<Application-Component app_id="Weblogic on Windows" name="WebLogic AS"
category="J2EE Server" ci_type="weblogicas" vendor="bea_systems_ltd">

     <process name="java.exe" main-process="true"
ports="weblogic,weblogicSSL,None" cmdline="-Dweblogic.Name=" >

               <service-endpoint-name>weblogic</service-endpoint-name>

          </process>

</Application-Component>
```

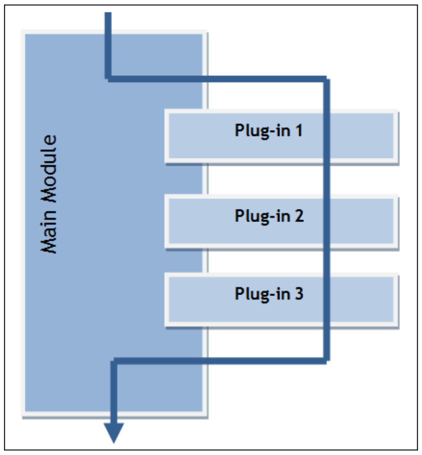
Plug-Ins

Application Signatures plug-ins are ways of dynamically adding or removing functionality . Usually plug-ins are independent of each other and perform a single task.

In Universal Discovery, there is always a flow of execution similar to the following:

- Discovery job starts
- Job performs some activity (i.e. executes commands)
- Job formats results as a vector of Object State Holders
- Job returns results and the execution ends

To add plug-ins to this job, the job at some point should pass control to the plug-ins and allow them to affect the results. This can be seen in the following diagram, where the arrow represents execution flow:



In general, for this mechanism to work at the point where the main module passes control to plug-ins, the answers to the following questions must be known:

- What plug-ins exist in the system?
- What plug-ins, from the whole set of plug-ins, should run along with this module?
- What is the order of plug-ins?

- Where is the plug-in’s code, and how can it be instantiated?
- How does a plug-in pass its results to main module?

Plug-Ins Framework Overview

AutoDiscoveryContent package includes the python library **plugins.py**. This library contains the following classes that are used by the plug-ins feature:

- **Plugin**

  This is a base class for plug-ins. You should extend it to create a new plug-in. It contains the following methods:

  - isApplicable(context) – Method that is called by the plug-ins' framework, where the plug-in has a chance to perform a runtime check against a passed data in order to verify that required conditions are met. For example, in Application Signatures you can verify that a found application has information about a process with a specific name, and this process has a non-empty command line. Method should return True(1) if requirements are met. Implementation of this method is optional.
  - process(context) - Main method of the plug-in; where the main functionality should reside.

- **PluginContext**

  This class represents an object that allows sharing data between the main module and a plug-in, and between plug-ins. The main module can pass relevant data and support objects to the plug-in, such as a Discovery Framework object, and initialized clients. In turn, the plug-in can use these objects for its work and store the results in the same context. Both methods of the Plugin class - **isApplicable()** and **process()** - accept single parameter context.

  Note: The base class **PluginContext** does not have any methods out-of-the-box, since such methods are mostly defined by the data that plug-ins want to use and may differ from module to module. If you who want to add support for plug-ins to a module, you must create the appropriate context class.

- **PluginEngine**

  This is a main class that encapsulates all work with plug-ins. Its purpose is to find appropriate plug-ins, instantiate them, and run them when the client code passes control to it. The client module should create this object and use it without being concerned how the plug-in mechanisms are implemented. This class has one public method:

- process(context, filter) – Main method of the plug-ins engine. In this method, engine forms a chain of plug-ins using the provided filter object. (See "PluginFilter", below.) For each such plug-in, it calls the isApplicable() method while passing context. If the plug-in returns **true**, the plug-in's method process() is called with the same context. This way the context is passed from plug-in to plug-in.

- **PluginFilter**

  This represents a class encapsulating the logic of filtering plug-ins: selecting only the required plug-ins from the pool of all available plug-ins. PluginFilter defines asingle method:

  filterPlugins(pluginDescriptors) – method that accepts the list of plug-in descriptors(described further) and returns the list of only those descriptors that satisfy some criteria. This method is called from PluginEngine.

- **QualifyingFilter**

  This is an implementation of **PluginFilter**, where the decision whether some plug-in should be included in the chain, is based on qualifiers defined in the plug-in descriptor and qualifiers of the filter itself. (See "Qualifiers" on page 692.) This class has the additional method: **addQualifier(type, value)** which adds a qualifier to this filter with a specific type and value.

Plug-In Packages Overview

Plug-ins are organized into plug-in packages. Each such package is a collection of plug-ins that usually have a common theme. For example: all discover SQL database version by shell. Usually the code for plug-ins is in one python module (a .py file) but it is possible to use multiple scripts if required.

Each plug-in package contains the following files:

- package configuration file
- one or more python scripts containing code for the plug-ins (one subclass of Plugin class per plug-in)

Plug-in Package Configuration File

The plug-in package configuration file is an XML file with meta information describing the plug-ins. For example:

db_versions.package.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<package parserClassName="com.hp.ucmdb.discovery.
library.communication.downloader.cfgfiles.PluginsPackageConfigFile">
    <plugins>
      <plugin id="mysql_version_by_shell">
         <name>MySql version by shell</name>
         <description>Sets MySQL version attribute for discovered
          MySQL Server CI</description>
         <module>plugins_appsignature_dbversion_by_shell</module>
         <class>MySQLVersionShellPlugin</class>
         <qualifiers>
           <qualifier type="application">MySQL DB</qualifier>
           <qualifier type="protocol">ntadmin</qualifier>
           <qualifier type="protocol">ssh</qualifier>
           <qualifier type="protocol">uda</qualifier>
           <qualifier type="protocol">telnet</qualifier>
         </qualifiers>
         <dependencies>
           <module>file_ver_lib</module>
           <module>mysql_version_by_shell</module>
         </dependencies>
       </plugin>
     ...
     </plugins>
</package>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<package parserClassName="com.hp.ucmdb.discovery.
library.communication.downloader.cfgfiles.PluginsPackageConfigFile">
    <plugins>
      <plugin id="mysql_version_by_shell">
         <name>MySql version by shell</name>
         <description>Sets MySQL version attribute for discovered
          MySQL Server CI</description>
         <module>plugins_appsignature_dbversion_by_shell</module>
         <class>MySQLVersionShellPlugin</class>
         <qualifiers>
           <qualifier type="application">MySQL DB</qualifier>
           <qualifier type="protocol">ntadmin</qualifier>
           <qualifier type="protocol">ssh</qualifier>
           <qualifier type="protocol">uda</qualifier>
           <qualifier type="protocol">telnet</qualifier>
         </qualifiers>
         <dependencies>
           <module>file_ver_lib</module>
           <module>mysql_version_by_shell</module>
         </dependencies>
```

```
        </plugin>
      ...
      </plugins>
</package>
```

The format of this configuration file is as follows:

- Attribute **id** of the **plugin** element defines the unique identification string for this plug-in. It is required to uniquely maintain the scope in all deployed packages.
- Element **name** contains the user-friendly name of this plug-in, which may appear in the UI.
- Element **description** contains a user-friendly description of the plug-in.
- Element **module** defines the name of python module (.py file) with the code of the plug-in.
- Element **class** defines the name of class that extends **Plugin** class, and which the plug-ins engine tries to instantiate.
- Elements **qualifier** defines all qualifiers of plug-in; each such element has the attribute **type**, which is a type of qualifier.
- Elements **module** enclosed in **dependencies** elements, defines the modules that should be loaded before loading the module of the plugin, and which are used by this plugin module.

Plug-In Python Module

The plug-ins python module is a regular python script that contains the code for plug-ins. Each plug-in should extend the **Plugin** class from **plugins.py**. You can have more than one plug-in in one python script, and you can have more than one script in one plug-ins package. Also, you are not limited in adding your own methods or classes.

Taking the "Plug-in Package Configuration File", the corresponding script follows:

plugins_appsignature_dbversion_by_shell.py

```
#coding=utf-8
import re
import sys
import logger
import mysql_version_by_shell
from file_ver_lib import getLinuxFileVer
from plugins import Plugin


class MySQLVersionShellPlugin(Plugin):
     """
         Plugin set MySQL version by shell, depends on OS type.
     """

     def __init__(self):
          Plugin.__init__(self)
          self.__client = None
          self.__process = None
          self.__isWinOs = None
          self.__cmd = None

     def isApplicable(self, context):
          self.__client = context.client
          try:
            if self.__client.isWinOs():
               self.__isWinOs = 1
               self.__process = context.application.getProcess
               ('mysqld-nt.exe')
               if not self.__process:
               self.__process = context.application.getProcess
               ('mysqld.exe')
         else:
               self.__process = context.application.getProcess
               ('mysqld')
         if self.__process:
               return 1
       except:
         logger.errorException(sys.exc_info()[1])

   def process(self, context):
      applicationOsh = context.application.getOsh()
      mysql_version_by_shell.setVersion(applicationOsh, self.__
      process.executablePath, self.__client)
...
class MySQLVersionShellPlugin(Plugin):
     """
         Plugin set MySQL version by shell, depends on OS type.
     """

     def __init__(self):
          Plugin.__init__(self)
          self.__client = None
          self.__process = None
          self.__isWinOs = None
```

```
           self.__cmd = None

      def isApplicable(self, context):
           self.__client = context.client
           try:
             if self.__client.isWinOs():
                self.__isWinOs = 1
                self.__process = context.application.getProcess
                ('mysqld-nt.exe')
                if not self.__process:
                self.__process = context.application.getProcess
                ('mysqld.exe')
          else:
                self.__process = context.application.getProcess
                ('mysqld')
          if self.__process:
                return 1
        except:
          logger.errorException(sys.exc_info()[1])

    def process(self, context):
       applicationOsh = context.application.getOsh()
       mysql_version_by_shell.setVersion(applicationOsh, self.__
       process.executablePath, self.__client)
...
```

Notice that script name and class name correspond to data in the configuration file.

Qualifiers

Qualifiers is the way to specify meta information for plug-in, which others can query and use. For example, you can say that a plug-in can run only on Windows, or is using a shell protocol. Qualifiers are like tags, but they contain two bits of information: **qualifier type** and **qualifier value**. Qualifiers are not limited in any way, so it is possible to use arbitrary textual values.

We use the qualifiers mechanism to select which plug-ins should run at a particular time. We create the QualifyingFilter instance and specify qualifiers on that filter. This filter selects only those plug-ins which specify qualifiers of the same type and the same values, or do not have qualifier of that type at all.

For example, in the "Plug-in Package Configuration File" on page 688, there are 2 qualifiers:

- The qualifier **application** specifies that the plug-in should run only for applications with the name “MySQL DB”.

- The qualifier **protocol** specifies that the plug-in works with the following protocols: NTCMD, SSH, UDA, and Telnet.

Qualifiers information is accessible from the plug-in configuration file before the plug-in is even instantiated. This enables the static filtering of plug-ins.

How to Create a New Plug-In

1. You must become familiar with specific implementations of the **PluginContext** class in order to know what data a plug-in passes on. For Application Signatures, refer to **applications.py** where the **ApplicationSignatureContext** class is located.

2. Create a new python module and add an implementation of the **Plugin** class there. In particular, you should write the code for methods **isApplicable()** and **process()** which accept context instance. The results of plug-in work should be saved to context.

3. Create a new plug-ins package configuration file with meta information about the plug-in; you should specify there information such as the python module name, class name , and so on. See "Application Signatures Configuration File" on page 679.

   Note: In particular, if necessary, you should specify the qualifiers for your plug-in which will be used to decide whether this plug-in should be run. You should know which qualifiers are actually used. For example, in Application Signatures, only 'application' and ‘protocol’ qualifiers are specified.

4. Add the newly created file to a discovery package - either new or existing.

Plug-In Limitations

The downside of qualifiers and pluggability is that it can be difficult to determine how many plug-ins exist, and which of them run for any particular discovered RunningSoftware.

Chapter 51: Host Resources and Applications by PowerShell Discovery

This chapter includes:

Overview

Windows PowerShell is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language built on top of, and integrated with, the .NET Framework. PowerShell provides full access to COM and WMI, enabling administrators to perform administrative tasks on both local and remote Windows systems.

How to Discover Host Resources and Applications by PowerShell

The following steps describe how to discover host resources and applications by PowerShell.

1. **Prerequisites - Set up protocol credentials**

   This discovery solution is based on the PowerShell protocol. The corresponding credentials must be filled in order to use it.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Before starting the discovery ensure that PowerShell v2.0 is installed on the Data Flow Probe machine.

2. **Run the discovery**

   To discover the topology:

   a. Run the **Range IPs by ICMP** or **Range IPs by nmap** job to discover the Windows system IP addresses.

   b. Run the **Host Connection by Powershell** job to discover how Windows connects with the PowerShell agent and networking topology.

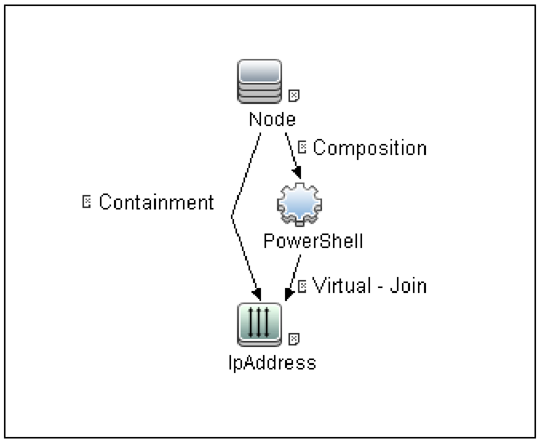
   c. Run the **Host Resources by PowerShell** and **Host Applications by PowerShell** jobs to discover the host resources topology.

      For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Host Resources and Applications by PowerShell Job

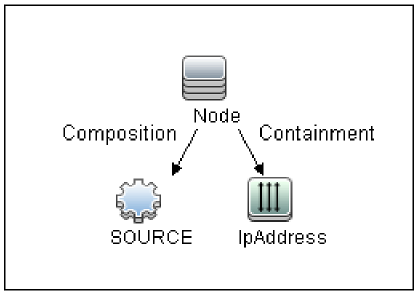
This section includes:

Trigger Query



Adapter

- Input CIT: PowerShell
- Input TQL Query:



Discovered CITs

- CPU
- FileSystem

- FileSystemExport
- IIS Application Pool
- InstalledSoftware
- IpAddress
- IpServiceEndpoint
- Node
- OS User
- Process
- RunningSoftware
- WindowsService
- ClientServer relationship
- Composition relationship
- Containment relationship
- Dependency relationship
- Realization relationship
- Usage relationship

Chapter 52: Host Resources and Applications Discovery

This chapter includes:

Overview

The **Hosts and Resources** module discovers resources that exist on a host (for example, Disk, CPU, Users) as well as applications that run on that host. The module also discovers the relationships between the application and the relevant processes, the appropriate services, and the relevant IP Service Endpoint (port).

The **Host Resources by Shell/SNMP/WMI** and **Host Applications by Shell/SNMP/WMI** jobs:

- Discover the TCP connections of the discovered machines, using Shell or SNMP.
- Store the information in the Data Flow Probe-dedicated `netflow` database.
- Query the Data Flow Probe database for TCP information.

The **Host Resources by Shell** and the **Host Applications by Shell** jobs also gather connectivity information (either by running `netstat` commands or the **lsof** command).

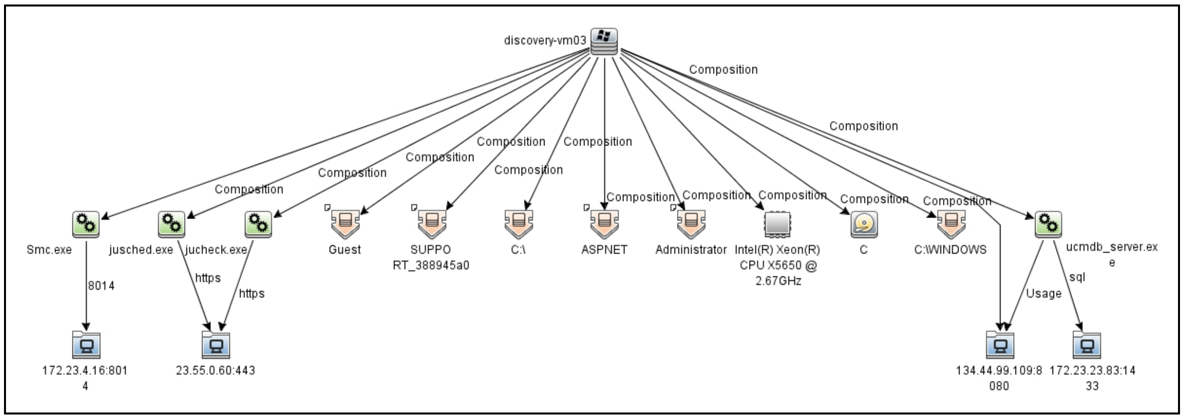
The relationships between processes and the relevant IP Service Endpoint (server port) can be discovered on Windows 2003 and Windows XP, SunOS, Hewlett-Packard UniX (HP-UX), AIX, and Linux operating systems.

For the HP-UX and AIX machines, you should install `lsof` software, which can be downloaded from the Internet from, for example, **http://www.netadmintools.com/html/lsof.man.html**. You can install **lsof** software also on SunOS. If you do not, the **pfiles** software that is installed on SunOS is used.

Note: Process to process (**P2P**) discovery is the name given to the discovery of processes running on hosts in the environment.

Topology

Note: For a list of discovered CITs, see "Discovered CITs" on page 709.


discovery-vm03
Composition
Smc.exe
jusched.exe
jucheck.exe
Guest
SUPPORT_388945a0
C:\
ASPNET
Administrator
Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
C
C:\WINDOWS
ucmdb_server.exe
8014
https
https
Usage
sql
172.23.4.16:8014
23.55.0.60:443
134.44.99.109:8080
172.23.23.83:1433

How to Discover Host Resources and Applications

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   To run this module, define the following protocols:

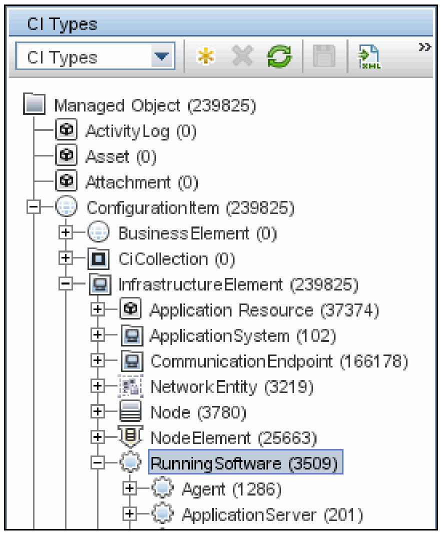
   - NTCMD protocol
   - SNMP protocol
   - SSH protocol
   - Telnet protocol
   - WMI protocol

   Users do not need root permissions, but do need the appropriate credentials to enable connecting to the remote machines and running the relevant commands.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   Verify that the CMDB already contains the Agent and Shell CITs: **Modeling > CI Type Manager**. Search for **RunningSoftware**, and verify that Agent and Shell are present:


CI Types

CI Types

- Managed Object (239825)
  - ActivityLog (0)
  - Asset (0)
  - Attachment (0)
  - ConfigurationItem (239825)
    - BusinessElement (0)
    - CiCollection (0)
    - InfrastructureElement (239825)
      - Application Resource (37374)
      - ApplicationSystem (102)
      - CommunicationEndpoint (166178)
      - NetworkEntity (3219)
      - Node (3780)
      - NodeElement (25663)
      - RunningSoftware (3509)
        - Agent (1286)
        - ApplicationServer (201)

3. **Run the Host Resources by Shell/SNMP/WMI and Host Applications by Shell/SNMP/WMI discovery**

   In the Universal Discovery window, activate the relevant **Host Resources by Shell/SNMP/WMI** and **Host Applications by Shell/SNMP/WMI** jobs.

   The former jobs discover resources that exist on a node (for example, Disk, CPU, Users) and the latter discover applications that run on that host. (See "Application Signatures" on page 666.) The jobs are scheduled to run every day.

How to Revert to Previous Method of Discovering Installed Software

The Host Applications by WMI job discovers installed software that is installed using the WMI Windows Installer Provider.

If the software is not installed with the Windows Installer, you must use the previous mechanism to discover the software.

To revert to the previous discovery mechanism for this job:

1. Access the Host Resources and Applications by WMI adapter: **Adapter Management > Host_Resources_By_WMI > Adapters > WMI_HR_All**.

2. In the **Adapter Definition** tab, locate the **Adapter Parameters** pane.

3. Double-click the **discoverInstalledSoftwareByOldMechanism** parameter to change the default value from **false** to **true**.

4. Save the change.

   A warning message is added to the communication log.

Host Resources and Applications Discovery

This section includes:

Job Threads

Each job is run using multiple threads. You can define a maximum number of threads that can be used concurrently when running a job. If you leave the box empty, the Data Flow Probe's default threading value is used (8).

The default value is defined in **DataFlowProbe.properties** in the **defaultMaxJobThreads** parameter.

- **regularPoolThreads.** The maximum number of worker threads allocated to the multi-threaded activity (the default is 50).
- **priorityPoolThreads.** The maximum number of priority worker threads (the default is 20).

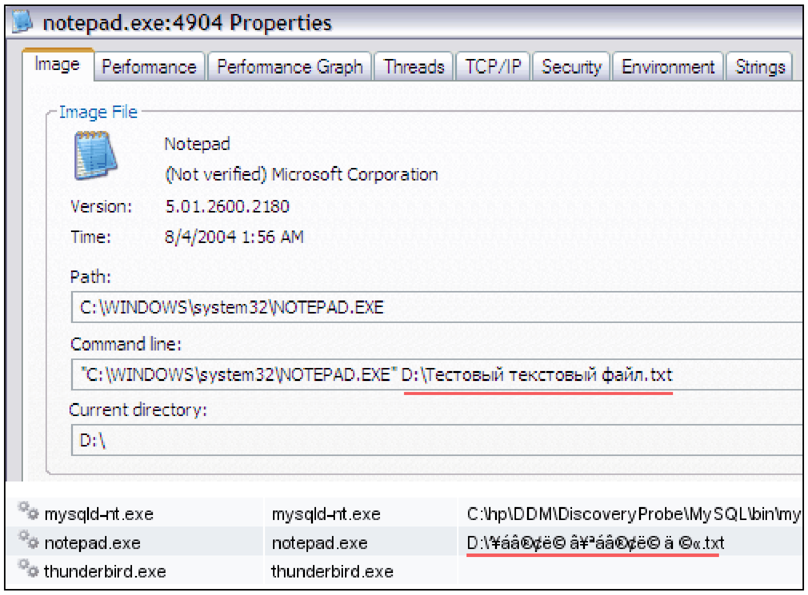
Note:

- The number of actual threads should never be higher than `regularPoolThreads + priorityPoolThreads`.

- The jobs in this module require a permanent connection to the Data Flow Probe's internal database. Therefore, these jobs are limited to a maximum number of 20 concurrent threads (which is the maximum number of concurrent connections permitted to the internal database).
- For details on the Max. Threads field, see "Execution Options Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Locale-Based Processes

Discovery detects the locale used on a remote machine by searching for known keywords, adjusting the encoding, and using the correct regular expressions and strings. However, output may include characters in more than one language, in which case the characters may become corrupted. For example, in the following graphic, the command line uses a text file with Russian file name on an English Windows machine:



Notepad
(Not verified) Microsoft Corporation
Version: 5.01.2600.2180
Time: 8/4/2004 1:56 AM
Path: C:\WINDOWS\system32\NOTEPAD.EXE
Command line: "C:\WINDOWS\system32\NOTEPAD.EXE" D:\Тестовый текстовый файл.txt
Current directory: D:\

mysqld-nt.exe	mysqld-nt.exe	C:\hp\DDM\DiscoveryProbe\MySQL\bin\my
notepad.exe	notepad.exe	D:\¥áâ®¢ë© â¥ªáâ®¢ë© ä ©«.txt
thunderbird.exe	thunderbird.exe	

To prevent character corruption, Discovery uses a **wmic** command that saves the file in UTF-16 encoding. This is controlled by the **useIntermediateFileForWmic** parameter in the **globalSettings.xml** file (**Adapter Management > AutoDiscoveryContent > Configuration Files**). **True**: the parameter is enabled. The default value is **false**.

Adapter Parameters for the Host Resources by Shell and Host Applications by Shell Jobs

For details, see "Adapter Parameters Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Parameter	Description
P2PServerPorts	Only processes connected to these ports (as client or server) are discovered, together with this port. This parameter can include a number or a known name. You separate entries with commas. An asterisk (*****) signifies all ports. The default value is *****.
discoverCPUs	Enable to discover CPUs. **Default:** True.
discoverDisks	Enable to discover disks. **Default:** True.
discoverFcHBAs	Enable to discover Fibre Channel HBAs. **Default:** False. Supported platforms and protocols: • Windows over Shell-based and WMI protocols • SunOs over Shell-based protocol • HP-UX over Shell-based protocol
discoverInstalledSoftware	Perform software discovery. If set to true, all installed software (with or without a signature) is reported. **Default:** False. **Note:** Setting this parameter to true may adversely affect performance on the destination machine.
discoverMemory	Enable to discover memory. **Default:** True.

Parameter	Description
discoverModules	Specifies whether to perform module discovery. **Default:** False. **Note:** Applicable to Host Resource by SNMP job only.
discoverP2P	Specifies whether to report TCP connections for running processes. • True. TCP connections for running processes are reported. • False. TCP connections for running processes are not reported. **Default**: True.
discoverProcesses	• **False**: Only processes that are related to specified running software are discovered. (The running software is specified in the **applicationsSignature.xml** file.) • **True**. All processes are discovered. **Default**: False.
discoverRunningSW	Specifies whether or not to discover Running Software. **Default:** False.
discoverServices.	• **False**: services are not reported. • **True**. All services are discovered. **Default**: False.
discoverShares	**True**: Shared resources are discovered, and **FileSystemExport** CITs are created. **Default**: True.
discoverUsers	Enable to discover users. **Note:** For Windows machines, only local users are discovered. **Default:** True.

Parameter	Description
filterP2PProcessesByName (formerly `filterProcessesByName`)	The names of the processes that are not reported. **Default:** system,svchost.exe,lsass.exe,System Idle Process. To prevent P2P running, enter an asterisk (*****) as the value.
ignoreP2PLocalConnections	**False**: P2P discovery does not ignore local connections. That is, when a client and server are installed on the same host and the client-server relationship connects between them, P2P discovery should report this relationship.
lsofPath	The path to the `lsof` command that enables process communication discovery on UNIX machines. The default value is **/usr/local/bin/lsof,lsof,/bin/lsof**.
useLSOF	**True:** Discovery tries to use lsof utility to discover port-to-process mappings on UNIX machines. **Default:** True.
useNetstatOnly	Specifies whether or not to run additional commands (LSOF) or to use the netstat command only. **Default:** False .
wmicPath	Specifies the location of the wmic.exe utility on Windows. **Default:** %SystemRoot%\system32\wbem\.

Adapter Parameters for the Host Resources by SNMP and Host Applications by SNMP Jobs

For definitions of the parameters, see "Adapter Parameters for the Host Resources by Shell and Host Applications by Shell Jobs" on page 706.

Adapter Parameters for the Host Resources by WMI and Host Applications by WMI Jobs

For definitions of the parameters, see "Adapter Parameters for the Host Resources by Shell and Host Applications by Shell Jobs" on page 706.

TCP Discovery

The Client/server relationship. When checking connections between two destinations (IP and port pairs), DFM uses the following logic to decide which side is the server and which the client (descending, in order of importance):

- If one of the ports is a listening port (that is, is marked as listening in the `port_process` table), then this port is a server port.
- If one of the ports is used by a process that is known to be a server process, then this port is the server port.
- If a local port is not listening and the remote side has not yet been processed (TCP discovery has not yet run on the remote side), it is assumed that the remote port is the server port.
- If neither port is listening and none of the processes is known to be a server process, DFM does not report P2P connectivity.

Discovered CITs

- hardware_board
- environmental_sensor
- fan
- power_supply
- printer_toner
- printer_tray

Alternatively, to view all discovered CITs, select a specific adapter in the **Resources** pane.

For details, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Note: To view the topology, see "Topology" on page 699.

Scripts Used

- snmp_model_finder.py
- snmp_model_discovery.py

Note: Applicable to Host Resource by SNMP job only.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Host Resources and Applications discovery.

- To discover processes and software running on a Solaris machine, verify that the **/usr/ucb/ps** utility is installed on the Solaris machine.
- Discovery of processes that have names with spaces is not supported on UNIX machines.
- Discovery of non-English content brought by ssh and telnet clients from UNIX machines is not supported.
- The installation date of installed software is not reported if the software was installed under a non-English-locale user.
- When DFM discovers installed software by WMI, and the software does not include a defined name, DFM does not report the software entity to the CMDB.
- The jobs **Host Resource by SNMP** and **Host Applications By SNMP** produce corrupted multibyte characters if the name or description of host resources (for example: processes, windows services, users, installed software) contains multibyte characters.

Chapter 53: IBM i by Eview Discovery

This chapter includes:

Overview

The IBM i by Eview discovery is a full iSeries Agent based discovery for iSeries (AS400) servers. It uses the EView Technology iSeries client and Agent to perform the discovery on the iSeries system. The EView Agent is installed on the iSeries node to execute the discovery.

Note: Refer to the EView 400 iSeries documentation for installation instructions.

The iSeries EView Client is installed on each probe that will be used to do IBM i by Eview Discovery jobs.

Areas of Discovery

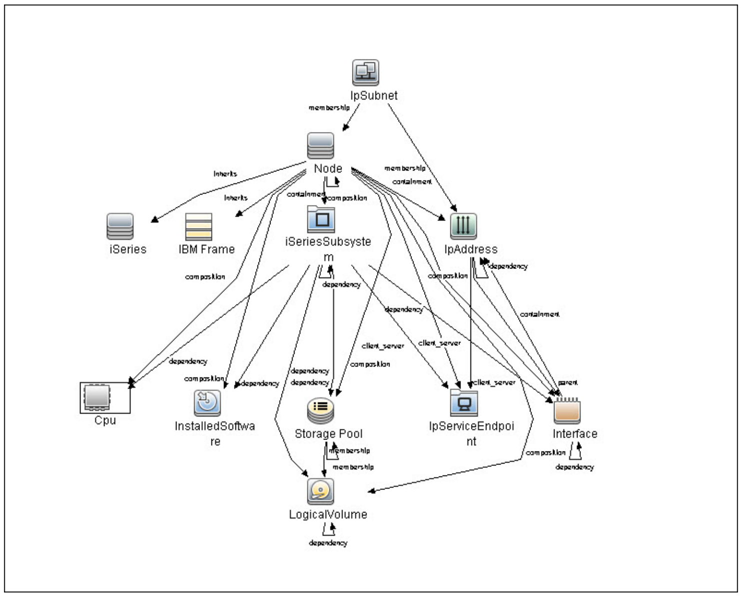
- **IBM i Resources**
  - Local Storage with ASPs
  - Memory
  - Lpars
  - CPUs
  - Network Connectivity
  - Installed Software
  - Selected System Values
  - Subsystems
  - Active Jobs
- **IBM i Objects**
  - Job Queues
  - Output Queues
  - Libraries
  - Program Objects

Supported Versions

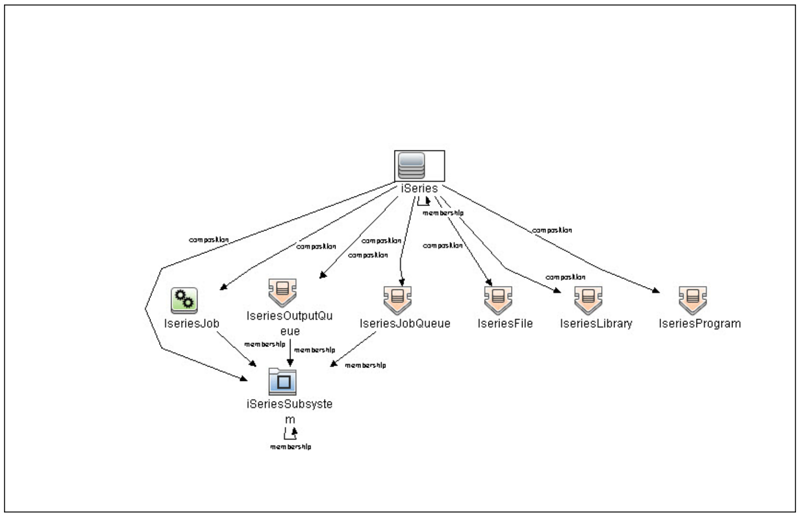
UCMDB Version	iSeries Version
9.x	OS/400 releases V5R1M0 and above

Topology

IBM i Resources



IBM i Objects



Discovery Mechanism

The discovery jobs use EView 400 Client and Agent. When activated, the discovery script uses the EView 400 client installed on the probe. The EView 400 client is accessed as a local shell.

The EView 400 client sends the commands issued by the script to the EView 400 agent running on the iSeries node. These commands are OS/400 and EView Agent commands. The result of the command execution is returned to the client, and then passed on to the calling script.

How to Discover iSeries

This task describes how to discover iSeries CIs using the EView Client and Agent.

1. **Prerequisites**

   Install **EView Agent** on the iSeries side, and **EView Client** on the DFM probe side. For instructions, refer to the EView 400 Discovery Installation Guide.

2. **Run the Discovery**

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **IBM i Connection** job to discover the target iSeries host .

   c. Run the **IBM i Resources** job to discover resource information from the iSeries lpar, such as Cpus, Memory, Auxiliary Storage Pools and Disks, Subsystems, and Network Connectivity.

   d. Run the **IBM i Objects** job to discover object information from the iSeries lpar, such as queues, jobs, program objects, and libraries.

IBM i Connection Job

This section includes details about the job.

Input CIT

• Probe

Used Scripts

- eview400_connection.py
- eview400_lib.py
- file_mon_utils.py
- file_ver_lib.py

Discovered CITs

- composition
- containment
- eview
- ip_address
- iSeries

Parameters

Parameter	Description
EViewInstallationFolder	The installation root directory of the EView client on the probe server.
debugMode	This may be set to **true** or **false**. If set to **true**, it enables detailed logging in the probe's debug log. **Default**: false

IBM i Objects Job

This section includes details about the job.

Trigger TQL

Input CIT: EView

Trigger Parameters

- ApplicationPath ${SOURCE.application_path:NA}
- LparName ${HOST.name}
- NodeName ${SOURCE.discovered_product_name}

Used Scripts

- eview400_objects.py
- eview400_lib.py

Discovered CITs

- composition
- iSeries
- iseries_job
- iseries_jobqueue
- iseries_library
- iseries_outputqueue
- iseries_program
- iseriessubsystem
- membership

Parameters

Parameter	Description
commandTimeout	The timeout value (in seconds) after which the command issued against the EView agent will timeout. **Default**: 60
debugMode	This may be set to **true** or **false**. If set to **true**, it enables detailed logging in the probe's debug log. **Default**: false
discover_Jobs	This may be set to **true** or **false**. If set to **true**, the job will discover the Active Jobs on the iSeries lpar. **Default**: false
discover_Library	This may be set to **true** or **false**. If set to **true**, the job will discover ISeries Library Objects. **Default**: true
discover_Program	This may be set to **true** or **false**. if set to **true**, the job will discover iSeries Program Objects. **Default**: false **Note:** Discovery of program objects is a time consuming job. Therefore, if setting this parameter to **true**, it is recommended to increase the value of the **commandTimeout** parameter.
discover_Queue	This may be set to **true** or **false**. if set to **true**, the job will discover the Queues (Job, Output). **Default**: true

IBM i Resources Job

This section includes details about the job.

Trigger TQL Query

Input CIT: EView

Trigger Parameters

- ApplicationPath ${SOURCE.application_path:NA}
- LparName ${HOST.name}
- NodeName ${SOURCE.discovered_product_name}

Used Scripts

- eview400_resources.py
- eview400_lib.py

Discovered CITs

- client_server
- composition
- containment
- cpu
- dependency
- IBM Frame
- installed_software
- interface

- ip_address
- ip_service_endpoint
- ip_subnet
- iSeries
- iseriessubsystem
- logical_volume
- Membership
- Node
- parent
- storagepool

Parameters

Parameter	Description
commandTimeout	The timeout value (in seconds) after which the command issued against the EView agent will timeout. **Default**: 60
debugMode	This may be set to **true** or **false**. If set to **true**, it enables detailed logging in the probe's debug log. **Default**: false
discover_ASP	This may be set to **true** or **false**. If set to **true**, the job will discover Auxillary Storage Pools and Disk Units. **Default**: false
discover_CPUs	This may be set to **true** or **false**. If set to **true**, the job will discover iSeries LPAR CPU CIs. **Default**: true

Parameter	Description
discover_Network	This may be set to **true** or **false**. If set to **true**, the job will discover ISeries Interface CIs. **Default**: true
discover_Software	This may be set to **true** or **false**. if set to **true**, the job will discover iSeries Installed Software CIs. **Default**: false
discover_Subsystems	This may be set to **true** or **false**. if set to **true**, the job will discover iSeries Subsystem CIs. **Default**: true
discover_TCP_UDP	This may be set to **true** or **false**. if set to **true**, the job will discover iSeries LPAR TCP ports and connectivity and UDP ports. **Default**: false

Inventory Discovery

Inventory Discovery, which also includes Infrastructure discovery, determines which devices are in your network and gathers information about each of them. It also serves as the foundation for the other modules of discovery.

Inventory Discovery is typically managed and run based on Management Zones, using wizard-driven activities. For information about Inventory Discovery activities, see Inventory Discovery Activity in the *HP UCMDB Universal Discovery Content Guide - Discovery Activities* document. For other information, see the section about Inventory Discovery in the *HP Universal CMDB Data Flow Management Guide*.

Chapter 54: NetApp Filer Discovery

This chapter includes:

Overview

HP Universal CMDB can retrieve NetApp network attached storage (NAS) information directly from NetApp Filers. Discovery involves synchronizing devices, topology, and hierarchy of storage infrastructure in the UCMDB database (CMDB). This enables change management and impact analysis across all business services mapped in UCMDB from a storage point of view.

The discovery involves a UCMDB initiated discovery on the NetApp Filer WebService API. The discovery also synchronizes physical relationships between various hardware, and logical relationships between logical volumes and hardware devices, to enable end-to-end mapping of the storage infrastructure.

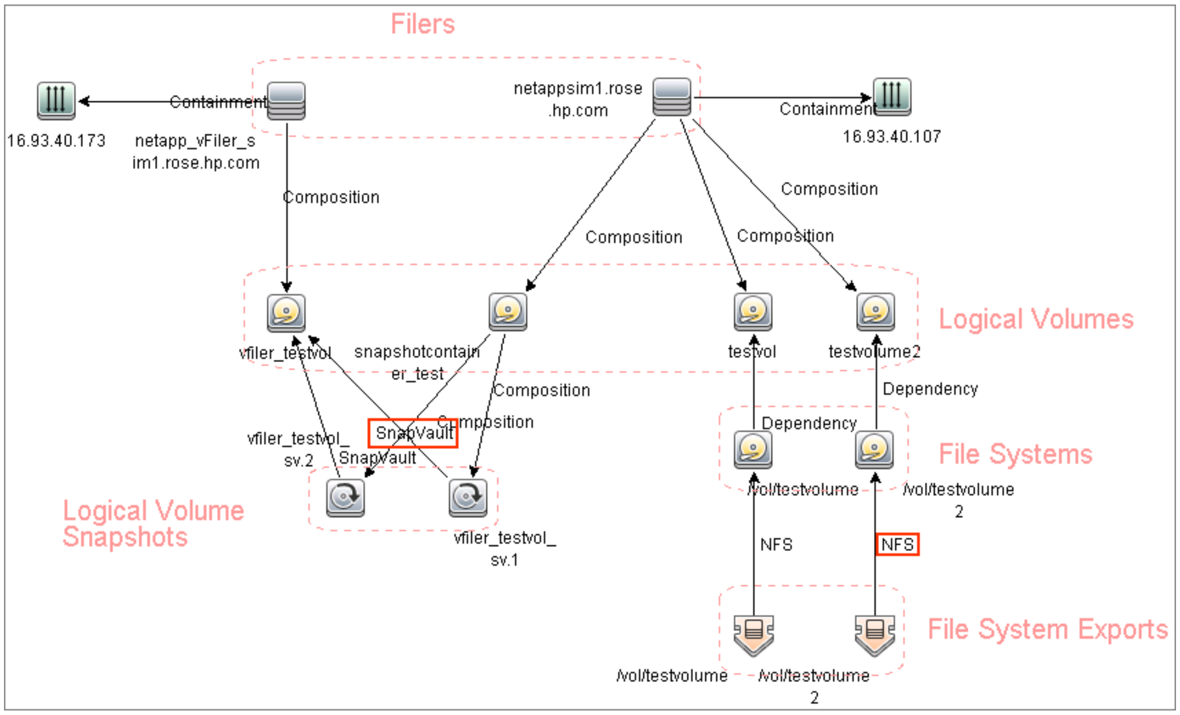
Supported Versions

This discovery supports NetApp Data ONTAP 7.2.x, 7.3.x, and 8.x with installed ONTAP SDK 3.5.1.

Topology

The following image displays the topology of the NetApp Filer discovery with sample output:

Note: For a list of discovered CITs, see "Discovered CITs" on page 729


Filers
Logical Volumes
File Systems
Logical Volume Snapshots
File System Exports
16.93.40.173
Containment
netapp_vFiler_sim1.rose.hp.com
netappsim1.rose.hp.com
Containment
16.93.40.107
Composition
Composition
Composition
Composition
vfiler_testvol
snapshotcontainer_test
testvol
testvolume2
Composition
Composition
Dependency
Dependency
vfiler_testvol_sv.2
SnapVault
SnapVault
vfiler_testvol_sv.1
/vol/testvolume
/vol/testvolume2
NFS
NFS
/vol/testvolume
/vol/testvolume2

How to Discover NetApp Filers

This task describes how to discover NetApp Filers.

1. **Prerequisite - Set up protocol credentials**

   This discovery includes the NetApp protocol for NetApp WebServices. To use the NetApp protocol, configure the appropriate credentials and port to the NetApp WebService API. The discovery uses the NetApp ONTAP SDK to get information from NetApp Filers.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Permissions**

   Note: For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Ensure the user has the appropriate permissions on the Netapp Filer system to run the following discovery commands:

Command	Description
`login-http-admin`	Required permission for the discovery. You cannot authenticate or login to a NetApp Filer using the Netapp WebServices API without it.
`api-system-get-info`	Get appliance details including CPU and backplane information. (Head information in a sysconfig -a command). I/O information is not included.
`api-system-get-ontapi-version`	Required to Get current ONTAPI major and minor versions.
`api-ipspace-list-info`	Get information about ipspaces including IP addresses and relevant IP details. (Requires **vfiler** license.)
`options-get`	Get values for optional parameters.

Command	Description

`api-volume-list-info-iter-start`
`api-volume-list-info-iter-next`
`api-volume-list-info-iter-end`
	Get details on volumes in the appliance.

`api-snapshot-list-info`	Get details on snapshots for a specified volume.

`api-snapvault-<SnapvaultLevel>`
`-relationship-status-list-iter-start`
`api-snapvault-<SnapvaultLevel>`
`-relationship-status-list-iter-next`
	Get snapvault details from the appliance. <SnapvaultLevel> can be either **primary** or **secondary** or both of these.

`api-cifs-share-list-iter-start`
`api-cifs-share-list-iter-next`
`api-cifs-share-list-iter-end`
	Get details on CIFS shares on this appliance. (Requires **cifs** license.)

`api-cifs-session-list-iter-start`
`api-cifs-session-list-iter-next`
`api-cifs-session-list-iter-end`
	Get details on CIFS sessions on this appliance. (Requires **cifs** license.)

`api-nfs-exportfs-list-rules`	Get details on NFS shares on this appliance.
`api-security-api-vfiler` `api-nfs-exportfs-list-rules-2`	Get details on vFilers.
`api-system-cli` `api-cli-ifconfig`	Get details on network interfaces.

3. **Run the discovery**

Note: For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Run the following jobs in the following order:

a. Run the **Range IPs by ICMP** job.

b. Run the **Host Connection by SNMP** job to identify NetApp Filers.

c. Run the **NetApp Filer by WebServices** job. For job details, see "NetApp Filer by WebServices Job" on the next page.

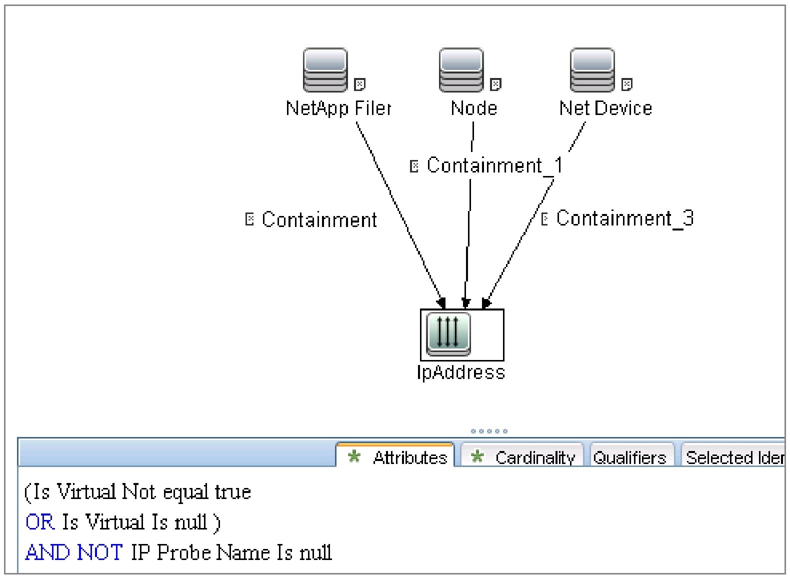
NetApp Filer by WebServices Job

The NetApp Filer discovery package is bundled in **NetAppFiler.zip**.

This section includes details about the job.

Trigger Query

This trigger TQL has the **include subtypes** option unselected for **Net Device** and **Node**, which will exclude IPs associated with CIs that are not NetApp Filers (such as Windows, UNIX, and so on).



(Is Virtual Not equal true
OR Is Virtual Is null)
AND NOT IP Probe Name Is null

Adapter

This job uses the **NetApp Filers by WebServices** adapter.

- Input query: None
- Adapter Parameters

Parameter	Description
getNetworkShareInfo	**True:** Network Shares discovery is performed. **False:** No Network Shares discovery is performed.

Parameter	Description
getSnapShotInfo	**True:** Logical Volume Snapshots discovery is performed. **False:** No Logical Volume Snapshots discovery is performed.
getSnapVaultInfo	**True:** SnapVault discovery is performed. **False:** No SnapVault discovery is performed.
chunksize	Maximum number of objects pulled from NetApp Operations Manager per SOAP call. To reduce the load on the NetApp Filer, set this parameter to a value lower than 1000 (default).
filerOptions	Discovers additional parameters and settings for theNetApp filer that are defined in the NetApp filer "Options" field. This parameter can contain comma-separated names of additional vFiler options to discover. Values of these options are stored in UCMDB in the **Options** attribute of NetApp Filer class. **Example:** nfs.tcp.recvwindowsize,nfs.tcp.xfersize,nfs.tcp.enable

Discovered CITs

- Composition
- Containment
- CPU
- Dependency
- FileSystem
- FileSystemExport
- Interface
- IpAddress
- Logical Volume Snapshot

- Logical Volume
- Membership
- Memory
- Netapp Filer
- Node
- Realization

Note: To view the topology, see "Topology" on page 723.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for NetApp Filer discovery.

The NetApp Filer by WebServices job does not identify vFilers. All of the vFilers resources are connected to the 'root' NetApp Filer.

Chapter 55: SMI-S Discovery

This chapter includes:

Overview

The Storage Networking Industry Association (SNIA) evolved and developed the Storage Management Initiative Specification (SMI-S) as a standard way of managing Storage Area Networks (SAN). The specification, which is ratified as an ISO standard, includes data accessible from a CIM server through a WEBM client.

This package discovers storage-related data using the CIM protocol.

Supported Versions

This discovery supports SMI-S version 1.50, TPD, CIMV2, EVA, LsiArray13, and EMC namespaces, and gives generic support for the CIM namespace.

Discovery Mechanism

Using the CIM protocol, the discovery connects to the target machine running storage management and the CIM agent. After connecting, the discovery counts the CIM classes, parses attributes, and maps to the UCMDB class model.

How to Discover SMI-S

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   The discovery uses the CIM protocol. You should define the CIM protocol entry, with port, namespace, and http/https transport.

   > **Note:** The CIM protocol supports http and https sub transports. The supported namespaces are
   >
   > - **root/tpd**
   > - **root/cimv2** (with limited support)

- **root/eva**
- **root/LsiArray13**
- **root/emc**

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the jobs**

   a. Run the **Range IPs by ICMP** job to discover the IP address of the server used by SMI-S.

   b. Run the **Storage Devices Connection by CIM** job to discover a proper credential and create a CIM CI.

   c. Run the **Storage Devices Topology by CIM** job to discover storage topology.

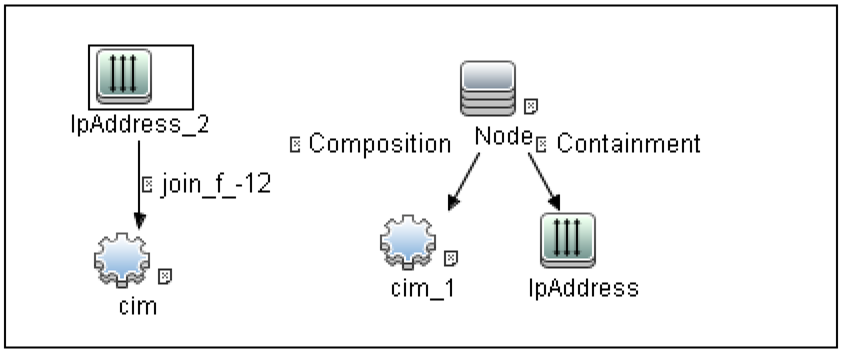
   For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Storage Devices Connection by CIM Job

Adapter

This job uses the **Storage Devices Connection by CIM** adapter.

Trigger TQL



Discovery Flow

The discovery checks to see if it is possible to retrieve instances of one of the CIM classes depending on the namespace:

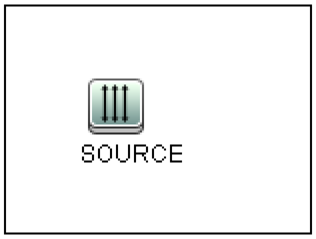
- Namespace: "root/cimv2" -> CIM_OrganizationalEntity
- Namespace: "root/tpd" -> TPD_StorageSystem
- Namespace: "root/eva" - > HPEVA_StorageSystem
- Namespace: "root/LsiArray13" -> LSISSI_StorageSystem
- Namespace: "root/emc" -> EMC_ComputerSystem

Storage Devices Connection by CIM Adapter

Input CIT

IP Address

Input TQL Query



Triggered CI Data

Name	Value
ip_domain	${SOURCE.routing_domain}
ip_address	${SOURCE.name}

Used Scripts

- cim.py
- entity.py
- smis_discoverer.py
- smis_connection.py
- cim_discover.py
- smis.py

Discovered CITs

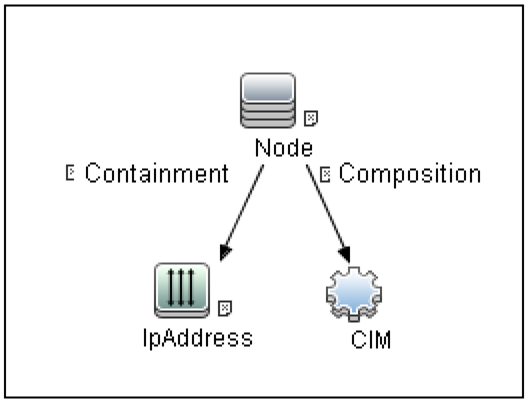
- CIM
- Composition
- Containment
- IpAddress
- Node

Storage Devices Topology by CIM Job

Adapter

This job uses the **Storage Devices Topology by CIM** adapter.

Trigger TQL Query



Node Name	Condition
IPAddress	NOT IP Probe Name Is null
IPAddress	NOT UcmdbRoutingDomain Is null
CIM	NOT Reference to the credentials dictionary entry Is null
CIM	CimCategory Contains Storage

Discovery Flow

In this discovery, the classes for those instances retrieved and parsed are listed for each namespace:

- **Namespace "root/tpd"**
  - TPD_AllocatedFromStoragePool
  - TPD_DynamicStoragePool
  - TPD_FCPort
  - TPD_NodeSystem
  - TPD_SCSIController
  - TPD_StoragePool
  - TPD_StorageSystem
  - TPD_StorageVolume
- **Namespace "root/cimv2"**
  - CIM_FCPort
  - CIM_NodeSystem
  - CIM_StoragePool
  - CIM_StorageSystem
  - CIM_StorageVolume
- **Namespace "root/eva"**
  - HPEVA_StoragePool
  - HPEVA_StorageSystem
  - HPEVA_DiskFCPort
  - HPEVA_StorageVolume
  - HPEVA_ProtocolControllerForVolume

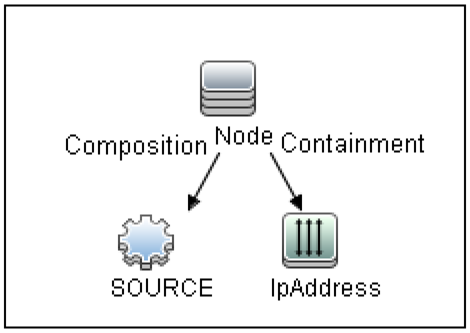
  - HPEVA_ViewProtocolController
  - HPEVA_StorageProcessorSystem
  - HPEVA_DiskExtent
- **Namespace "root/LsiArray13"**
  - **LSISSI_StoragePool**
  - **LSISSI_StorageSystem**
  - **LSISSI_FCPort**
  - **LSISSI_StorageVolume**
  - **LSISSI_ControllerCanister**
  - **LSISSI_ControllerFirmwareIdentity**
  - **LSISSI_StorageProcessorSystem**
  - **LSISSI_DiskExtent**
- **Namespace "root/emc"**
  - **EMC_StoragePool**
  - **EMC_StorageSystem**
  - **EMC_FCPort**
  - **EMC_StorageVolume**
  - **EMC_StorageSystemSoftwareIdentity**
  - **EMC_ArrayChassis**
  - **EMC_SCSIProtocolController**
  - **EMC_StorageProcessorSystem**
  - **EMC_DiskExtent**

Storage Devices Topology by CIM Adapter

Input CIT

cim

Input TQL Query



Node Name	Condition
IPAddress	NOT IP Probe Name Is null

Used Scripts

- cim.py
- entity.py
- smis.py
- smis_discoverer.py
- smis_topology.py
- cim_discover.py

Discovered CITs

- Composition

- Containment
- Fibre Channel Connect
- Fibre Channel Port
- Logical Volume
- Membership
- Node
- Storage Array
- Storage Pool

Part 9: Mainframe

Chapter 56: EView Agent Discovery

This chapter includes:

Overview

Many enterprise applications span mainframe and distributed (Linux/UNIX/Windows) environments. Sometimes the level of mainframe involvement is light (for example, only for backend database solutions), while at other times the mainframe can host more than the distributed side (for example, running through queues, middle-tier applications, and multiple mainframe subsystems).

The goal of HP Data Flow Management (DFM) is to properly map applications across the infrastructure, regardless of where those applications reside. There are normally three parts to mapping an application across the infrastructure:

1. Discovering the infrastructure
2. Discovering the application
3. Mapping the application dependencies

The current discovery solution covers the first two parts on the mainframe by discovering z/OS host and network resources, as well as applications such as DB2, IMS, CICS, and MQ.

The Mainframe by EView discovery is an agent-based discovery solution. It uses an application called **EView/390z Discovery for z/OS** to discover the Mainframe topology.

For more information about the discovery mechanism, see "Discovery Mechanism" on page 750.

To run the discovery, see "How to Discover Mainframe by EView" on page 749.

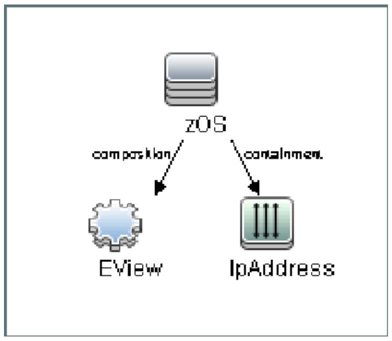
Supported Versions

Target Platform	Version
z/OS	1.8, 1.9, 1.10, 1.11, 1.12
DB2 for z/OS	8, 9
CICS	3.x, 4.x
WebSphere MQ on z/OS	6.0, 7.0
IMS	9+

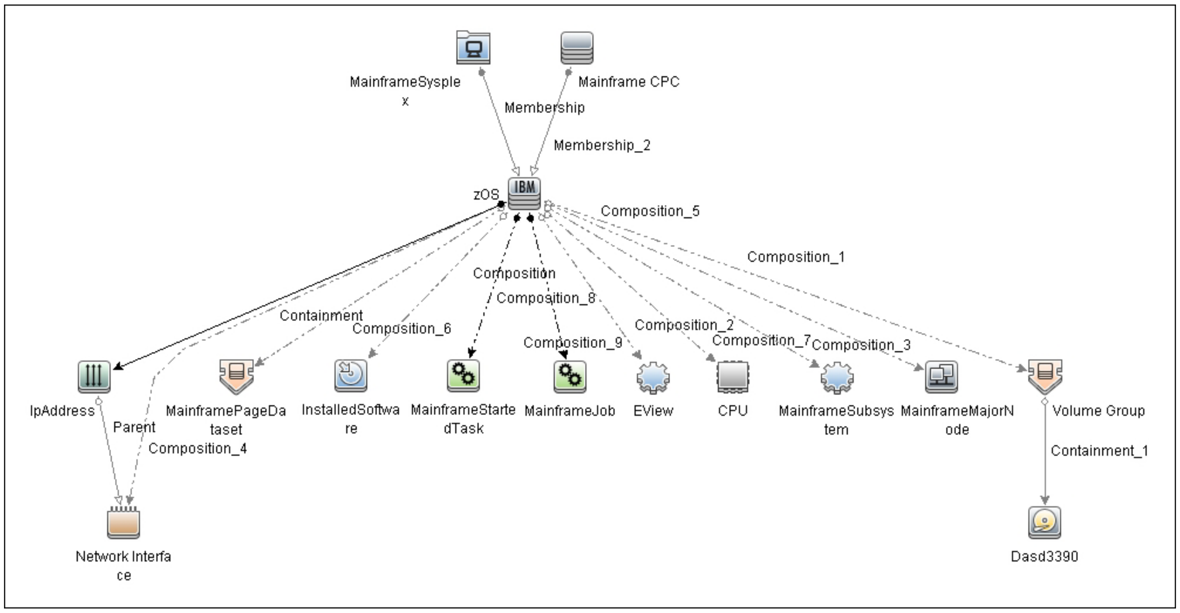
Topology

This section displays topology maps for the following jobs:

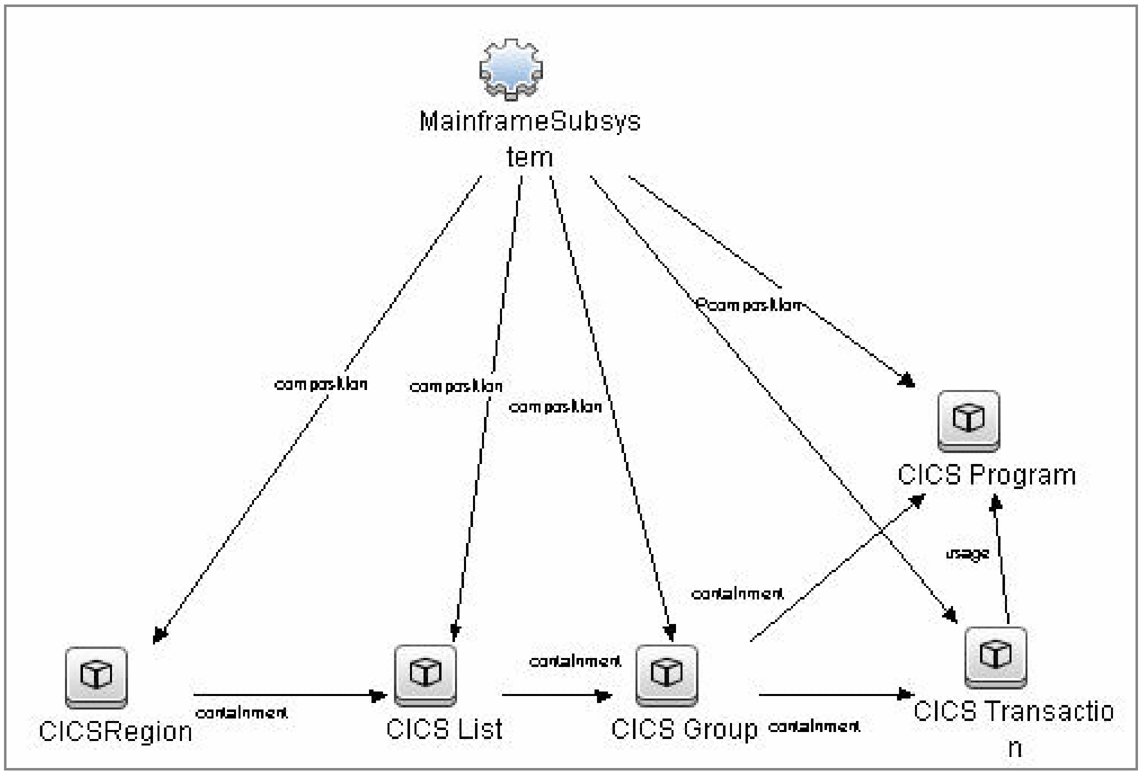
EView Connection



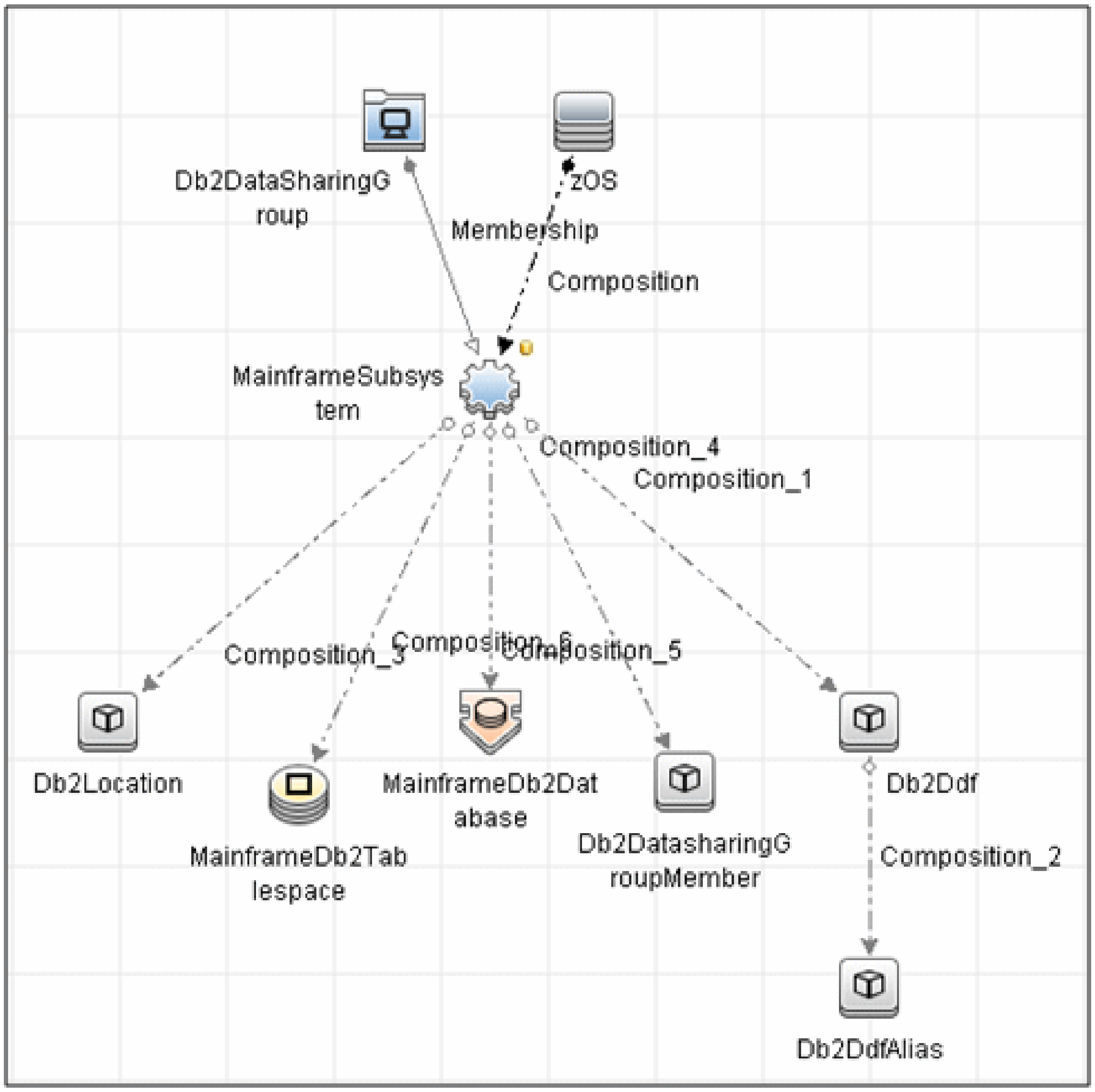
LPAR Resources by EView



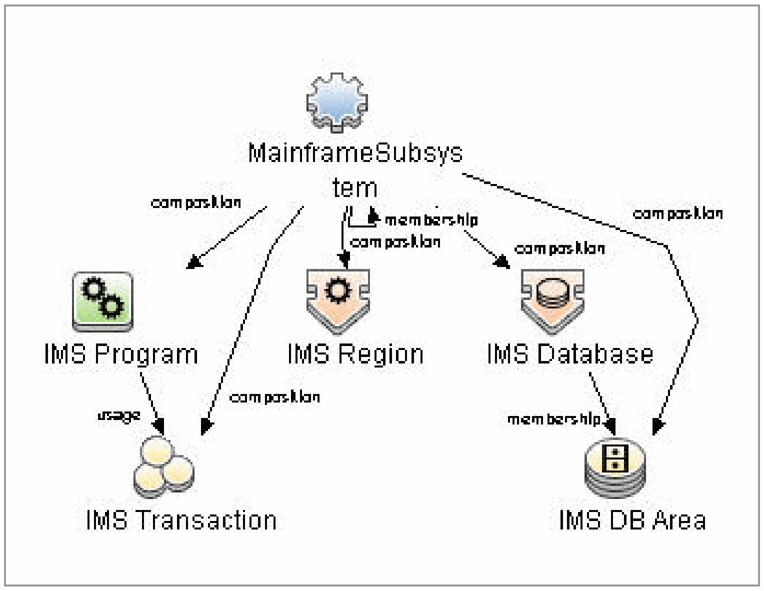
CICS by EView



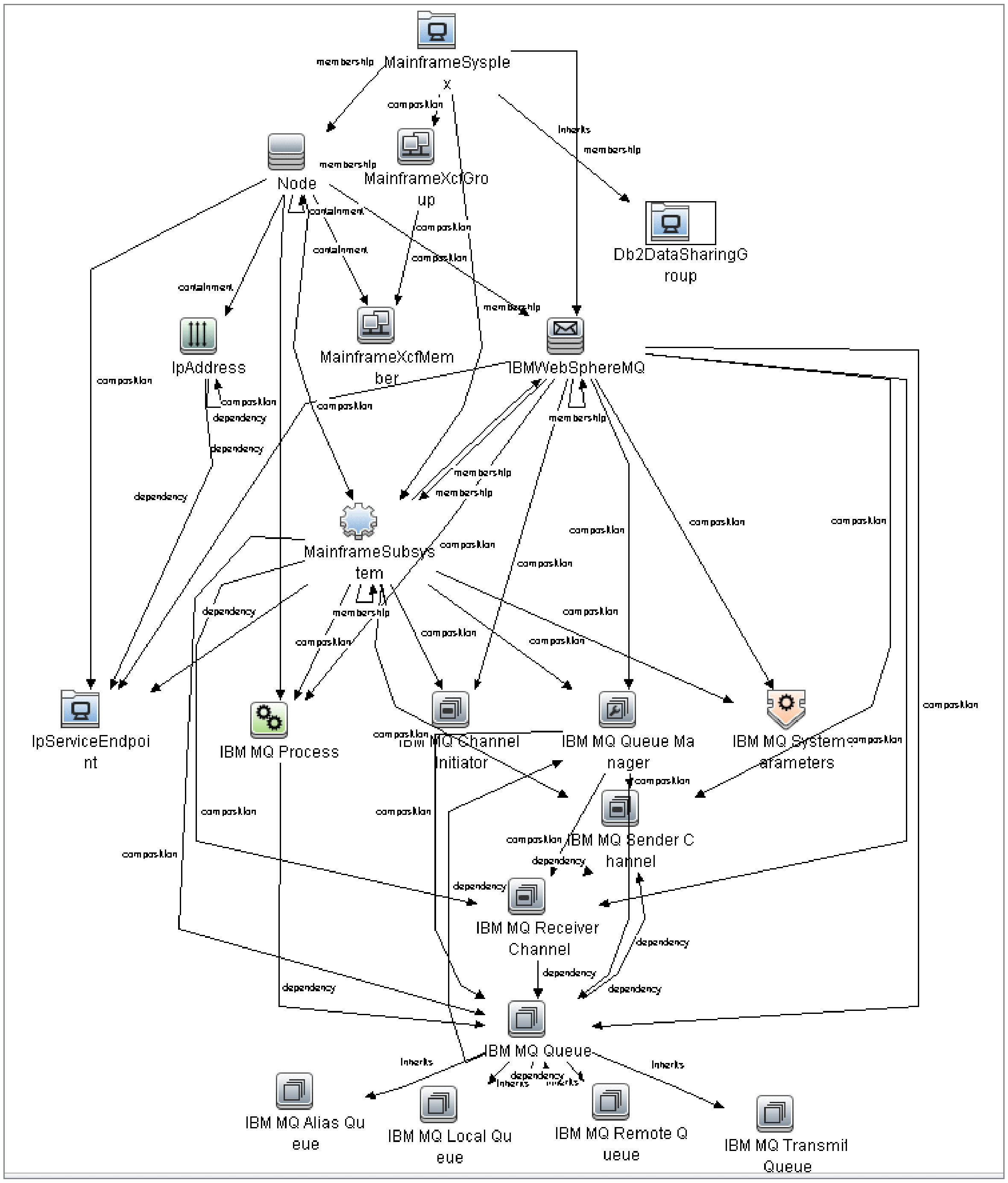
DB2 by EView


Db2DataSharingGroup
zOS
Membership
Composition
MainframeSubsystem
Composition_4
Composition_1
Composition_3
Composition_6
Composition_5
Db2Location
MainframeDb2Tablespace
MainframeDb2Database
Db2DatasharingGroupMember
Db2Ddf
Composition_2
Db2DdfAlias

IMS by EView


MainframeSubsystem
composition
membership
composition
composition
composition
IMS Program
IMS Region
IMS Database
usage
composition
membership
IMS Transaction
IMS DB Area

MQ by EView


MainframeSysplex
Node
MainframeXcfGroup
Db2DataSharingGroup
IpAddress
MainframeXcfMember
IBMWebSphereMQ
MainframeSubsystem
IpServiceEndpoint
IBM MQ Process
IBM MQ Channel Initiator
IBM MQ Queue Manager
IBM MQ System Parameters
IBM MQ Sender Channel
IBM MQ Receiver Channel
IBM MQ Queue
IBM MQ Alias Queue
IBM MQ Local Queue
IBM MQ Remote Queue
IBM MQ Transmit Queue
membership
composition
containment
dependency
inherits

How to Discover Mainframe by EView

The following steps describe Mainframe by EView discovery.

1. **Prerequisites**

   - Make sure that the EView/390z Agent (version 6.3 or later) is installed on every LPAR whose resources and applications have to be discovered.
   - Make sure that the EView/390z Discovery Client (version 6.3 or later) is installed on the same machine as the Data Flow Probe that will be used to discover the mainframe infrastructure.
   - Make sure that LPARs in the EView/390z Discovery Client are properly configured.
   - Make sure that all Security requirements have been set up for this discovery.

     For more information about these prerequisites, refer to the EView/390z Discovery for z/OS documentation:
     http://www.eview-tech.com/e390dldisc.php.

2. **Run the EView Connection job**

   Note: You must run this job before running any of the other Mainframe by EView discovery jobs.

   a. Configure the EView Connection discovery job's **EViewInstallationFolder** parameter by providing the absolute path to the EView/390z Discovery Client installation on the Data Flow Probe machine.

      For example:

      ```
      C:\EviewTechnology\EView390
      ```

   b. Activate the discovery job to discover the EView/390z Agent objects configured for every node in the EView/390z Discovery Client configuration on the Data Flow Probe machine.

3. **Run the discovery jobs**

   Activate the following jobs to discover the Mainframe topology:

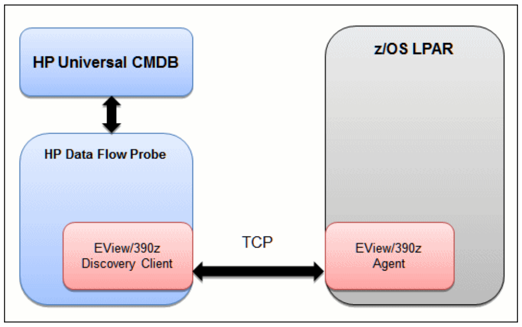
- Activate the **LPAR Resources by EView** job to discover the z/OS LPAR host and network resources. For details about this job, see "LPAR Resources by EView Job" on the next page.
- Activate the **CICS by EView** job to discover the CICS subsystem and its resources. For details about this job, see "CICS by EView Job" on page 754.
- Activate the **DB2 by EView** job to discover the DB2 subsystem and its resources. For details about this job, see "DB2 by EView Job" on page 755.
- Activate the **IMS by EView** job to discover the IMS subsystem and its resources. For details about this job, see "IMS by EView Job" on page 756.
- Activate the **MQ by EView** job to discover the MQ subsystem and its resources. For details about this job, see "MQ by EView Job" on page 757.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Discovery Mechanism

The Mainframe by EView discovery is an agent-based discovery solution. To discover infrastructure resources and applications on z/OS LPARs, an agent component must be deployed on every LPAR that has to be discovered.

A high-level architectural diagram for this discovery solution is illustrated in the following image:



The discovery process works as follows:

1. Connection job:

   a. The **EView Connection** job is the first job that discovers CIs for this discovery. It triggers against all the configured Probe Gateway CIs in the UCMDB.

   b. On the Data Flow Probe, the **eview_connection.py** discovery script first looks for the presence of the EView/390z Discovery Client in the pre-configured EView/390z Discovery Client installation path in the discovery job. It then looks for the z/OS LPAR nodes that have been configured in the EView/390z Discovery Client.

   c. For every configured z/OS LPAR node in the EView/390z Discovery Client, the discovery job creates an eview agent CI connected to a zOS CI along with a CI for its primary IP address.

2. Resource and application discovery jobs:

   a. The remaining jobs are all activated on the TQL query **eview_agent**, which invokes the job against all discovered eview agent CIs.

   b. The discovery scripts execute various MVS commands against the z/OS LPAR using the EView/390z Agent, parse the returned output, and create the relevant CI types.

For details on running the discovery, see "How to Discover Mainframe by EView" on page 749.

LPAR Resources by EView Job

This section includes details about the job.

Trigger Query

Trigger query name: **eview_agent**

Discovery Parameters

Parameter	Description
commandTimeout	Timeout value (in seconds) after which the command issued against the EView/390z Agent will timeout
maxCommandSize	Maximum size (in bytes) allocated for command output on the z/OS LPAR
debugMode	Set to true to enable detailed logging in the probe debug log
discover_CPUs	Looks for zOS LPAR CPU CIs

Parameter	Description
discover_Jobs	True/False flag indicating whether or not to discover the Address Spaces (Jobs, Started Tasks). **Default:** False
discover_ MajorNodes	Looks for zOS Major Node CIs
discover_ PageDatasets	Looks for zOS Page Dataset CIs
discover_ Software	Looks for zOS Installed Software CIs
discover_ Subsystems	Looks for zOS Subsystem CIs
discover_TCP_ UDP	Looks for z/OS LPAR TCP ports and connectivity and UDP ports
discover_DASD	Looks for z/OS Dasd Storage Devices and Storage Groups. **Default:** False **Note:** If set to True, you should increase the value of the command timeout parameters on the EView/390 client.
job_Regex	This parameter contains a UNIX style regular expression value to determine what jobs will be discovered. **Default:** * **Note:** If set to the default value, all jobs are discovered if **discover_Jobs** is set to **true**.

Note: To see a topology map of this discovery, see "LPAR Resources by EView" on page 745.

EView Connection Job

This section includes details about the job.

Trigger Query

Trigger query name: **probe**

Discovery Parameters

Parameter	Description
EViewInstallationFolder	Installation root directory of the EView/390z Discovery Client on the Data Flow Probe machine
EViewStartedTask	Started task name of the EView Agent (e.g. VP390)

Note: To see a topology map of this discovery, see "EView Connection" on page 744.

CICS by EView Job

This section includes details about the job.

Trigger Query

Trigger query name: **eview_agent**

Discovery Parameters

Parameter	Description
commandTimeout	Timeout value (in seconds) after which the command issued against the EView/390z Agent will timeout.
maxCommandSize	Maximum size (in bytes) allocated for command output on the z/OS LPAR.
debugMode	Set to true to enable detailed logging in the probe debug log.
discover_CICS_ Regions	Looks for CICS Regions and their detailed properties.
discover_CICS_ programs	True/False flag indicating whether or not to discover CICS programs and transactions. **Default:** False **Note:** If set to True, you should increase the value of the command timeout parameters on the EView/390 client.
exclude_ restricted_ programs	True/False flag indicating whether or not to discover IBM-supplied elements that are labeled 'RESTRICTED'. These elements are the standard operating components for the Vendor software packages. **Default:** True

Note: To see a topology map of this discovery, see "CICS by EView" on page 745.

DB2 by EView Job

This section includes details about the job.

Trigger Query

Trigger query name: **eview_agent**

Discovery Parameters

Parameter	Description
commandTimeout	Timeout value (in seconds) after which the command issued against the EView/390z Agent will timeout
maxCommandSize	Maximum size (in bytes) allocated for command output on the z/OS LPAR
debugMode	Set to true to enable detailed logging in the probe debug log
discover_DDF	Looks for z/OS DB2 Distributed Data Facility
discover_ DataSharingGroups	Looks for z/OS DB2 Distributed Datasharing Group
discover_Databases	Looks for z/OS DB2 Databases
discover_Locations	Looks for z/OS DB2 Locations
discover_ Tablespaces	Looks for z/OS DB2 Tablespaces

Note: To see a topology map of this discovery, see "DB2 by EView" on page 746.

IMS by EView Job

This section includes details about the job.

Trigger Query

Trigger query name: **eview_agent**

Discovery Parameters

Parameter	Description
commandTimeout	Timeout value (in seconds) after which the command issued against the EView/390z Agent will timeout.
debugMode	True/False flag. Set to true to enable detailed logging in the probe debug log.
maxCommandSize	Maximum size (in bytes) allocated for command output on the z/OS LPAR.
DiscoverIMSDB	True/False flag indicating whether or not to attempt to discover IMS Databases. **Default:** False
discover_ims_ programs	True /False flag indicating whether or not to discover IMS Programs and Transactions. **Default:** False

Note: To see a topology map of this discovery, see "IMS by EView" on page 747.

MQ by EView Job

This section includes details about the job.

Trigger Query

Trigger query name: **eview_agent**

Discovery Parameters

Parameter	**Description**
commandTimeout	Timeout value (in seconds) after which the command issued against the EView/390z Agent will timeout.
debugMode	True/False flag. Set to True to enable detailed logging in the probe debug log.
maxCommandSize	Maximum size (in bytes) allocated for command output on the z/OS LPAR.
Discover_remote_hosts	True/false flag indicating whether or not to attempt to discover hosts and queues on connected remote hosts. **Default:** False

Note: To see a topology map of this discovery, see "MQ by EView" on page 748.

Troubleshooting and Limitations

Troubleshooting Mainframe by EView discovery falls under two broad categories:

- Troubleshooting the UCMDB/DFM Mainframe discovery process:
  - Validating correct triggers for discovery jobs, checking invocation of discovery jobs, checking probe logs for troubleshooting information, and so on
  - Manually invoking commands against the z/OS LPAR using the EView/390z Discovery Client
  - Validating connectivity between the EView/390z Discovery Client and the EView/390z Agent
  - Checking that the commands can be issued successfully and valid responses are returned from the z/OS LPAR
- Troubleshooting the EView/390z Agent.

The discovery troubleshooting process almost always starts when a discovery process fails to correctly discover CIs and relationships. It is important then to determine whether the root-cause of the issue is with the UCMDB/DFM discovery process (jobs, triggers, adapters, scripts, and so on) or with EView/390z Discovery for z/OS. Some steps that can be helpful in this troubleshooting process are:

- Ensure that UCMDB/DFM processes/services are running as normal.
- Ensure that all the Mainframe discovery packages are correctly deployed and that the discovery jobs are properly configured.
- Ensure that the EView/390z Discovery Client (version 6.3 or later) and EView/390z Agent (version 6.3 or later) are installed. If earlier versions are installed, the discovery might fail.
- Ensure that the EView/390z Discovery Client is properly installed on the Data Flow Probe machine and its services are installed correctly and running.
- Ensure that the LPARs to be discovered are correctly configured in the EView/390z Discovery Client.
- Run the discovery job that is having issues and check the discovery logs for messages related to the invocation of jobs and execution of commands.
  - If there appears to be a problem with the invocation of discovery jobs, discovery script syntax errors, or CI reconciliation errors, troubleshoot them as you would any discovery process in

UCMDB.

- If the logs show that the discoveries are failing due to commands not being issued against the EView/390z Agent, identify the failing command from the probe debug log files, and manually try to invoke the relevant commands using the EView/390z Discovery Client. For more information, contact EView Technology Inc.'s customer support.

Part 10: Middleware > Java EE Application Servers

Chapter 57: Apache Tomcat Discovery

This chapter includes:

Overview

To discover Apache Tomcat, DFM parses the following configuration files:

- **server.xml**. This is the main Apache Tomcat configuration file that describes the components of the Tomcat installation, its architecture, and its topology. The file also contains the configuration for global resources.

  The following script fragment appears in the **server.xml** file and is the part used by the **Apache Tomcat by Shell** job to retrieve information for building the CIs:

  ```
  <Server port="8505" shutdown="SHUTDOWN">
    <GlobalNamingResources>
      <Resource name="jdbc/GlobalDS"
           type="javax.sql.DataSource"
           driverClassName="com.inet.ora.OraDriver"
           url="jdbc:inetora:labm3mam13:1521:UCMDB"
           maxActive="20" />
 </GlobalNamingResources>
 <Service name="Catalina">
   <Connector port="8580" protocol="HTTP/1.1"/>
   <Connector port="8509" protocol="AJP/1.3" />
   <Engine name="Catalina">
     <Host name="localhost" appBase="webapps">
       <Cluster">
           <Membership mcastAddr="228.0.0.4" mcastPort="45564"/>
       </Cluster>
     </Host>
     <Host name="grabinovic01" appBase="genadiwebapps">
           <Membership mcastAddr="228.0.0.4" mcastPort="45564"/>
          </Cluster>
        </Host>
      </Engine>
    </Service>
  </Server>
  ```

- **context.xml**. This file defines the application context configuration. Each installed application has a unique URL prefix. This file contains resource configurations for different scopes, depending on the file location.

- **web.xml**. This file defines the application configuration, for example, the application display name and the servlets used to process HTTP requests. Currently, DFM uses this file to retrieve the application display name.

Supported Versions

This discovery supports the following Apache Tomcat versions:

- 5
- 5.5
- 6.0

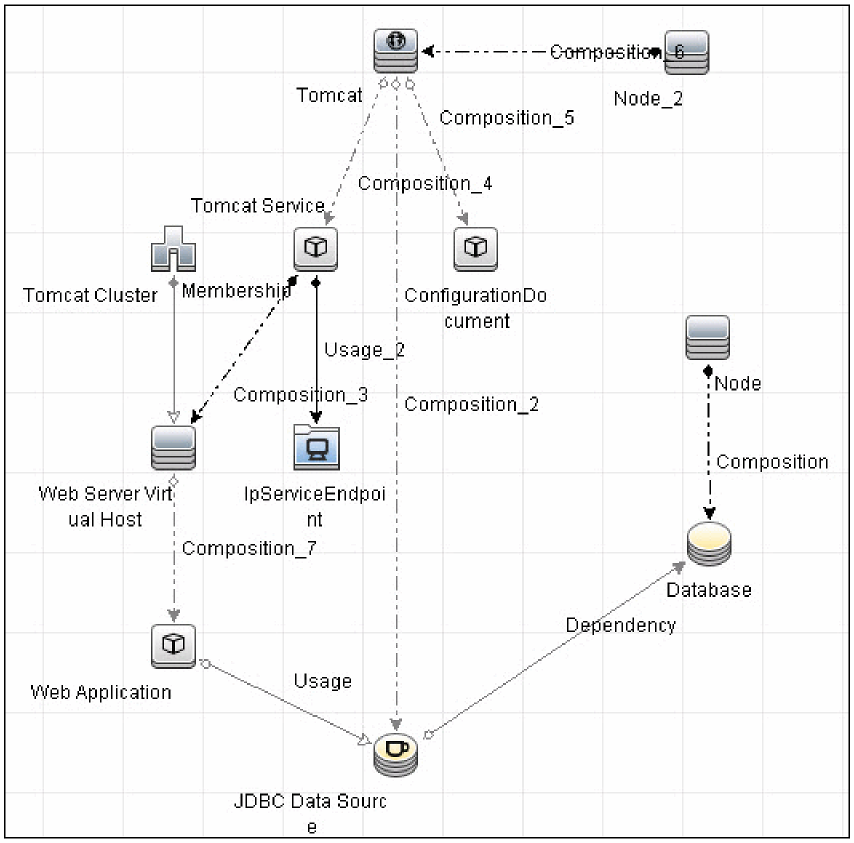
DFM discovers Tomcat running on the following operating systems:

- Windows
- UNIX
- Linux

Topology

The following image displays the topology of the Apache Tomcat discovery.

Note: For a list of discovered CITs, see "Apache Tomcat by Shell Job" on page 767.



How to Discover Apache Tomcat

This task describes how to discover the Apache Tomcat application and includes the following steps:

1. **Prerequisite - Set up network and protocol credentials**

   This discovery uses the following protocols:

   - NTCMD Protocol
   - SSH Protocol
   - Telnet Protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the Discovery**

   a. Run the **Range IPs by ICMP** job to discover IPs in the range where Tomcat is running.

   b. Run the **Host Connection by Shell** job to discover Shell agents.

   c. Run the **Host Applications by Shell** job to verify that an Apache Tomcat is running on the system, and to discover Tomcat-specific processes. If these processes are discovered, the job creates Tomcat CIs.

      The job searches for the **java.exe** (or **java**) process name, then searches in the command line for either the **-Dcatalina.home=** or **-Dcatalina.base=** substring. This substring includes the path to the Tomcat home directory. If this substring is not found, the job searches for a process name starting with **tomcat** and, from there, acquires the path to the home directory.

      The job then finds the absolute path to the Tomcat configuration file and adds this path as an attribute (**webserver_configfile**) to the Tomcat CI.

   d. Run the **Apache Tomcat by Shell** job. This job uses the Tomcat Trigger CI attribute to locate the configuration files that are discovered by the **Host Applications by Shell** job.

      For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Bugzilla, Wordpress, and MediaWiki

The following Web-based applications are discovered as part of the Apache and IIS discovery jobs. The following versions are supported:

Application	**Supported Version**
Bugzilla	3.x
Helpzilla	0.x
MediaWiki	1.15.x
Wordpress	2.5.x

To activate discovery:

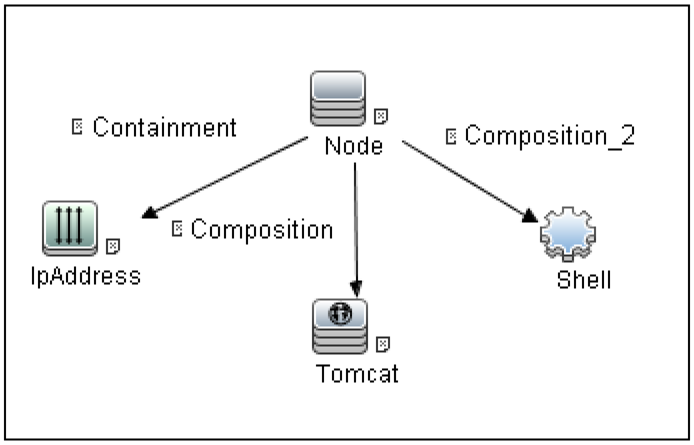
1. Run the **Host Connection by Shell** job to create Shell CITs.
2. Run any of the **Host Resources and Applications** jobs to gather information about processes running on the host.
3. Run the **Web Server by Shell** job to retrieve information about Apache and available Web applications deployed on the Apache server.

   The Web Application CIT:

   - **ID**. `webapplication`
   - **Parent CIT**. `application`
   - **Usage of the existing attribute**. `name`
   - **New attribute**. `type` (the type of application, for example, blog engine, wiki)

Apache Tomcat by Shell Job

Trigger Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
Shell	None
Node	None
Tomcat	None

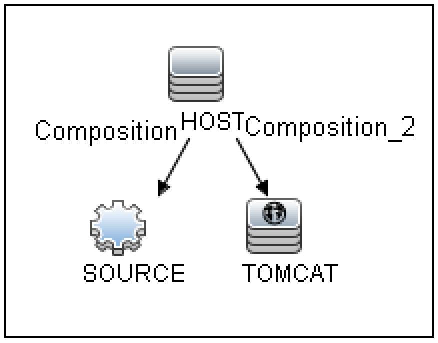
Adapter Information

This job uses the **ApacheTomcat_Topology** adapter.

Input CIT

Shell

Input TQL Query



Node Name	Condition
SOURCE	None
Tomcat	None
HOST	None

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
configfile	${TOMCAT.webserver_configfile}
credentialsId	${SOURCE.credentials_id}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}

Used Scripts

- tomcat_by_shell.py
- jdbc.py
- jdbc_url_parser.py

- db_builder.py
- db_platform.py
- db.py
- file_mon_utils.py
- file_ver_lib.py
- iteratortools.py
- entity.py

Discovered CITs

- Apache Tomcat
- Apache Tomcat Cluster
- Apache Tomcat Service
- Composition
- ConfigurationDocument
- Containment
- Database
- Dependency
- IpAddress
- IpServiceEndpoint
- JDBC Data Source
- Membership
- Node
- Usage

- Web Application
- Web Server Virtual Host

Note: To view the topology, see "Topology" on page 764.

Chapter 58: GlassFish Discovery

This chapter includes:

Overview

GlassFish is an open source application server based on the source code for Sun Java System Application Server Platform Edition 9 (from Sun Microsystems), and on the source code for TopLink (from Oracle). GlassFish supports all Java platform Enterprise Edition API specifications such as JDBC, RMI, e-mail, JMS, web services and XML, and details how to make them work with one another.

The GlassFish discovery process enables the user to discover a full topology, including J2EE applications, JDBC and JMS resources.

Supported Versions

Version	Supported	J2EE Version	JVM Version
GlassFish 2.1	Yes	J2EE 1.5	JVM 1.5
GlassFish 3.1	Yes	J2EE 1.6	JVM 1.6

How to Discover GlassFish Topology by Shell

This task describes how to discover GlassFish using Shell protocols. The GlassFish discovery process enables the user to discover a complete GlassFish topology including J2EE applications, JDBC and JMS resources. DFM first finds application servers based on the Shell protocol or endpoints (TCP Ports) and then discovers the GlassFish J2EE environment and components by Shell.

1. Prerequisites - Set up protocol credentials

   Discovery is done using the Shell protocol. One of the following credentials should be defined:

   - SSH
   - Telnet
   - NTCMD

2. Run the discovery

a. Run the **Range IPs by ICMP** job in order to discover the target IPs.

b. Run the **Host Connection by Shell** job in order to discover the target host and shell connectivity to it.

c. Run one of the two jobs:

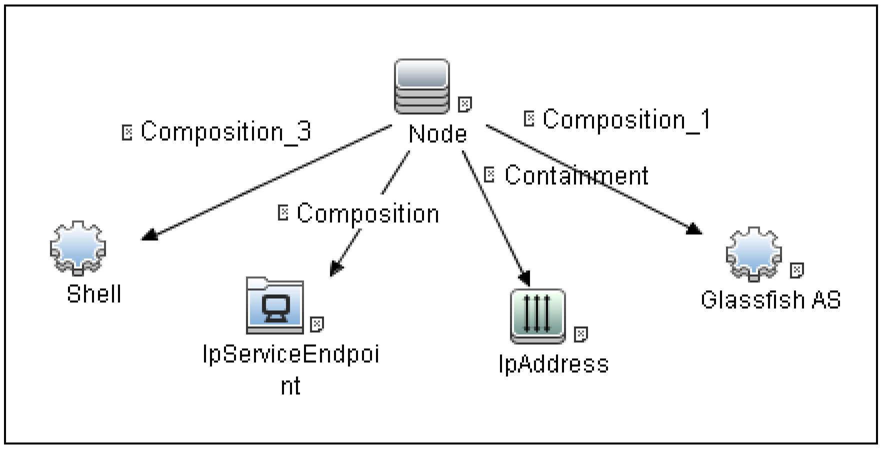
- **Host Applications by Shell** in order to discover applications of the target host, including running processes.
- **JEE TCP Ports** in order to discover service endpoint information.

3. Run the job **JEE Glassfish by Shell**.

JEE Glassfish by Shell Job

This section includes details about the job.

Trigger Query



Node Name	Condition
Glassfish AS	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndPoint	IpServiceName Equal glassfish OR ServiceNames Contains glassfish

Node Name	Condition
Node	None
Shell	NOT Reference to the credentials dictionary entry Is null

Adapter

This job uses the **Glassfish_By_Shell** adapter.

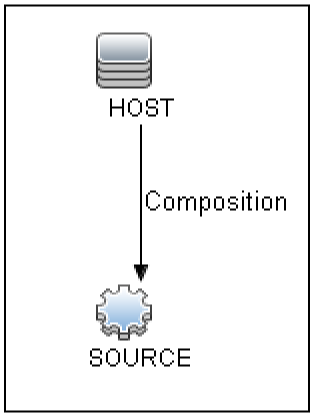
Glassfish_By_Shell Adapter

This section includes details about the adapter.

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}
protocol	${SOURCE.root_class}

Used Scripts

- connection.py
- db.py

- db_builder.py
- db_platform.py
- entity.py
- file_ver_lib.py
- glassfish.py
- glassfish_by_shell.py
- glassfish_discoverer.py
- iteratortools.py
- jdbc.py
- jdbc_url_parser.py
- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- process.py
- process_discoverer.py
- protocol.py

Discovered CITs

- Composition
- ConfigurationDocument

- Containment
- Database
- Dependency
- Deployed
- Glassfish AS
- IpAddress
- IpServiceEndPoint
- J2EE Cluster
- JDBC Data Source
- J2EE Domain
- J2EE Managed Object
- JEE Node
- Membership
- Node
- Usage
- Web Service

Global Configuration Files

globalSettings.xml

Parameters

- **reportAdminApps.** Enables/disables reporting of administrator applications if value is 'true'/'false'

Troubleshooting and Limitations

This section describes troubleshooting and limitations for GlassFish discovery.

- DFM can discover a J2EE application only when its .ear file is unzipped to a folder.
- Sometimes the command line of the GlassFish process is too large, so it does not fit in to the appropriate field in the probe database while running HRA discovery. In such a case:
  - Stop the probe
  - Open **%DatafFlowProbeHome%/tools/dbscripts/create_netlinks_db_tables.sql**
  - Change the size of **cmdline** for the Processes table from 4000 to 8000, or more if needed
  - Change the size of **cmdline** for the Applications table from 512 to 8000, or more if needed
  - Save the file
  - Run the **clearProbeData.bat** script
  - Start the probe

Chapter 59: JBoss Discovery

This chapter includes:

Overview

JBoss Application Server (or JBoss AS) is a free software/open-source Java EE-based application server developed by JBoss, now a division of Red Hat.

An important distinction for this class of software is that it not only implements a server that runs on Java, but it actually implements the Java EE part of Java. Because it is Java-based, the JBoss application server operates cross-platform: usable on any operating system that supports Java.

The JBoss discovery process enables you to discover a full JBoss topology including J2EE applications, JDBC, and JMS resources. DFM first finds JBoss servers based on the JMX protocol, then discovers the JBoss J2EE environment and components.

Supported Versions

- JBoss by JMX discovery: JBoss versions 3.x, 4.x, 5.x, 6.x, and 7.x
- JBoss by Shell discovery: JBoss versions 3.x, 4.x, 5.x, 6.x, and 7.x

How to Discover JEE JBoss by JMX

This task includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisites - Set up drivers" below
- "Run the discovery" on page 783

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the JBoss protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Set up drivers**

   Default JBoss drivers are included by default with the Probe installation. For details on the required

*.jar files, see "JBoss" in the *HP Universal CMDB Data Flow Management Guide*. The Probe installation includes JBoss drivers for versions 3.x and 4.x, but you can use your own drivers, if you prefer.

To update .jar files:

a. Copy the drivers to the correct version folder in the following location:

   ```
   C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
   \discoveryResources\j2ee\jboss\<version_folder>
   ```

   Note: There are errors in the commercial version of the JBoss 5.x client API (EAP). To discover EAP 5.x with authorization enabled, you must take the client drivers from a non-commercial version of 5.x.

b. Restart the Probe before running the DFM jobs.

   For example:

   To discover JBoss 5.x versions, you need to update the driver folder

   ```
   C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
   \discoveryResources\j2ee\jboss\5.x
   ```

   with the **jbossall-client.jar** file, including all dependencies declared in it.

   Required jars can be found in the **<JBOSS_5_BASE_DIR>/client/** folder.

   The **jbossall-client.jar** file contains a classpath reference to various client .jar files used by jboss client applications. Each of the .jar files in the following list must be available in the same directory as **jbossall-client.jar**. Otherwise they will not be found by the classloader.

   The classpath includes the following files:

   - commons-logging.jar
   - concurrent.jar
   - ejb3-persistence.jar
   - hibernate-annotations.jar
   - jboss-aop-client.jar

- jboss-appclient.jar
- jboss-aspect-jdk50-client.jar
- jboss-client.jar
- jboss-common-core.jar
- jboss-deployers-client-spi.jar
- jboss-deployers-client.jar
- jboss-deployers-core-spi.jar
- jboss-deployers-core.jar
- jboss-deployment.jar
- jboss-ejb3-common-client.jar
- jboss-ejb3-core-client.jar
- jboss-ejb3-ext-api.jar
- jboss-ejb3-proxy-client.jar
- jboss-ejb3-proxy-clustered-client.jar
- jboss-ejb3-security-client.jar
- jboss-ha-client.jar
- jboss-ha-legacy-client.jar
- jboss-iiop-client.jar
- jboss-integration.jar
- jboss-j2se.jar
- jboss-javaee.jar
- jboss-jsr77-client.jar

- jboss-logging-jdk.jar
- jboss-logging-log4j.jar
- jboss-logging-spi.jar
- jboss-main-client.jar
- jboss-mdr.jar
- jboss-messaging-client.jar
- jboss-remoting.jar
- jboss-security-spi.jar
- jboss-serialization.jar
- jboss-srp-client.jar
- jboss-system-client.jar
- jboss-system-jmx-client.jar
- jbosscx-client.jar
- jbosssx-as-client.jar
- jbosssx-client.jar
- jmx-client.jar
- jmx-invoker-adaptor-client.jar
- jnp-client.jar
- slf4j-api.jar
- slf4j-jboss-logging.jar
- xmlsec.jar

3. **Run the discovery**

   Run the following jobs in the following order:

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

- Run the **Range IPs by ICMP** job to discover the target IPs.
- Run the **JEE TCP Ports** job to discover service endpoint information. For job details, see "JEE TCP Ports Job" on page 786.
- Run the **JEE JBoss Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "JEE JBoss Connections by JMX Job" on page 790.
- Run the **JEE JBoss by JMX** job to perform a deep discovery of JBoss application server topology. For job details, see "JEE JBoss by JMX Job" on page 794.

How to Discover JEE JBoss by Shell

Note: This functionality is available as part of Content Pack 2.00 or later.

You can perform deep discovery of JBoss without having to enter JMX credentials for each server, and without having to define additional libraries (*.jar files). Instead, you use the regular Shell credentials.

Deep discovery enables you to discover the topology of J2EE application systems, that is, the components of an application and not just the application itself.

This task includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Run the discovery" below

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the Shell protocol. Define credentials for one of the following protocols:

   - NTCMD protocol
   - SSH protocol
   - Telnet protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Users do not need root permissions, but do need the appropriate credentials to enable connecting to the remote machines and running the relevant commands, such as **dir\ls** and **type\cat**.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **Host Connection by Shell** job to discover the target host and Shell connectivity to it.

   c. Run one of the two jobs:

- **Host Applications by Shell** to discover applications of the target host, including running processes.
- **JEE TCP Ports** to discover service endpoint information. For job details, see "JEE TCP Ports Job" below.

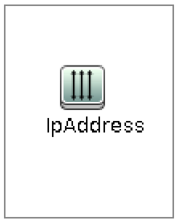
d. Run the **JEE JBoss by Shell** job. For job details, see "JEE JBoss by Shell Job" on page 797.

JEE TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	NOT IP Probe Name Is null

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host’s ports.

Name	Default Value	Description
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).
ports	**For JEE TCP Ports job:** • weblogic, weblogicSSL, websphere_jmx, rmi **For Database TCP Ports job:** • oracle, db2, sybase, sql, mysql **For SAP TCP Ports job:** • sap, sap_jmx, sap_http, sap_https **For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.

Name	Default Value	Description
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

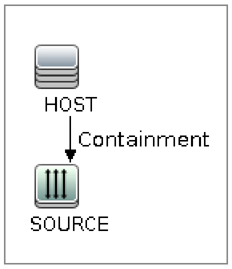
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

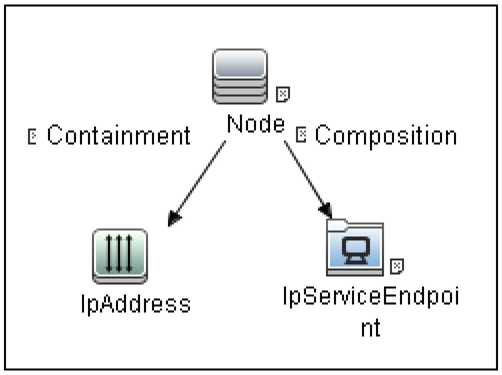
Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

JEE JBoss Connections by JMX Job

This section includes details about the job.

Trigger Query



Node Name	Condition
Node	None
IpServiceEndPoint	IpServiceName Equal rmi OR IpServiceName Equal jboss-port OR ServiceNames Contains rmi OR ServiceNames Contains jboss-port
IpAddress	NOT IP Probe Name Is null

Parameters

Name	Value	Description
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process
runInSeparateProcess	true	Should pattern run in separate thread

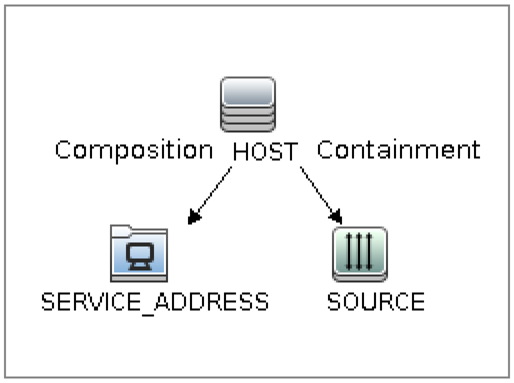
Adapter Information

This job uses the **JMX_J2EE_JBoss_Connection** adapter. This adapter discovers JBoss server instances based on the JMX protocol.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}
ports	${SERVICE_ADDRESS.network_port_number:NA}

Used Scripts

- connection.py
- db.py
- db_builder.py
- db_platform.py
- entity.py

- iteratortools.py
- j2eeutils.py
- jboss.py
- jboss_discoverer.py
- jdbc.py
- jdbc_url_parser.py
- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jmx.py
- JMX_J2EE_JBoss_Connection.py
- protocol.py

Discovered CITs

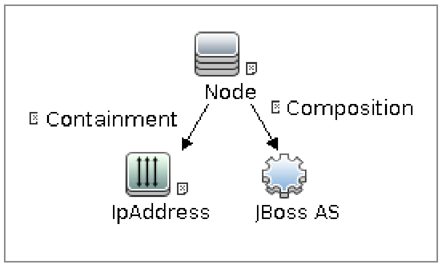
- Composition
- Containment
- IpAddress
- IpServiceEndPoint
- J2EE Domain
- JBoss AS
- JEE Node
- Membership

- Node
- Usage

JEE JBoss by JMX Job

This section includes details about the job.

Trigger Query



- Node Conditions

Node Name	Condition
Node	None
JBoss AS	`NOT Reference to the credentials dictionary entry Is null`
IpAddress	`NOT IP Probe Name Is null`

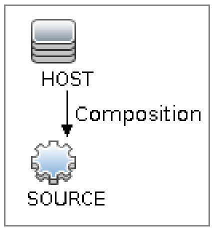
Job Parameters

Parameters are not overridden by default and use values from the adapter.

Adapter - JMX_J2EE_JBoss

This adapter discovers JBoss servers instances based on the JMX protocol.

- Input CIT: JBoss AS
- Input Query



- Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip:}
port	${SOURCE.application_port:}
servername	${SOURCE.name}
userName	${SOURCE.application_username:}
version	${SOURCE.application_version:}

- Used Scripts
  - connection.py
  - db.py
  - db_builder.py
  - db_platform.py
  - entity.py
  - iteratortools.py
  - j2eeutils.py
  - jboss.py
  - jboss_discoverer.py
  - jdbc.py
  - jdbc_ulr_parser.py
  - jdbcutils.py
  - jee.py
  - jee_connection.py

    - jee_discoverer.py
    - jms.py
    - jmx.py
    - JMX_J2EE_JBoss.py
    - protocol.py
- Global Configuration File: globalSettings.xml
- Parameters:

Name	Value	Description
discoverAppResources	true	Discover modules, ejbs and servlets if set to true.
discoverJMSResources	true	Discover jms providers and jms servers if set to true.
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process.
runInSeparateProcess	true	Whether the pattern should run in a separate thread.

Discovered CITs

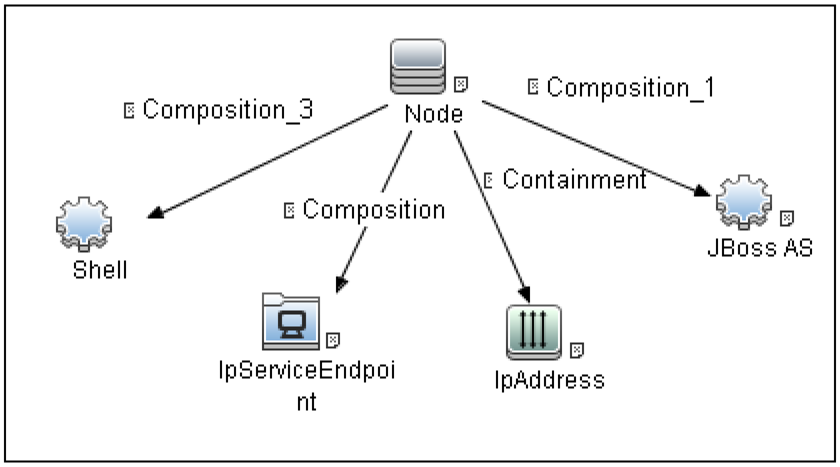
- Composition
- Containment
- ConfigurationDocument
- Database
- Dependency
- Deployed

- IpAddress
- IpServiceEndpoint
- J2EE Cluster
- J2EE Domain
- J2EE Managed Object
- JBoss AS
- JDBC Data Source
- JEE Node
- Membership
- Node
- Usage
- Web Service

JEE JBoss by Shell Job

This section includes details about the job.

Trigger Query



Node Name	Condition
Node	None
Shell	NOT Reference to the credentials dictionary entry Is null
JBoss AS	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndPoint	IpServiceName Equal rmi OR IpServiceName Equal jboss-port OR ServiceNames Contains rmi OR ServiceNames Contains jboss-port

Parameters

Parameters are not overridden by default and use values from the adapter.

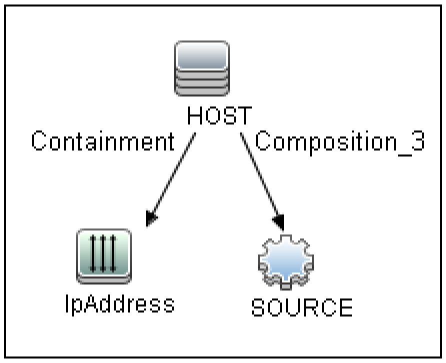
Adapter Information

This job uses the **JBoss_By_Shell** adapter.

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}
ip_address_list	${IpAddress.name}
Protocol	${SOURCE.root_class}

Used Scripts

- connection.py
- db.py
- db_builder.py
- db_platform.py
- entity.py
- file_ver_lib.py
- iteratortools.py
- jboss.py
- jboss_by_shell.py
- jboss_discoverer.py
- jdbc.py
- jdbc_url_parser.py
- jdbcutils.py
- jee.py

- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- process.py
- process_discoverer.py
- protocol.py

Global Configuration File

- globalSettings.xml

Parameters

None

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database
- Dependency
- Deployed
- IpAddress
- IpServiceEndPoint
- J2EE Cluster
- J2EE Domain

- J2EE Managed Object
- JBoss AS
- JDBC Data Source
- JEE Node
- Membership
- Node
- Usage
- Web Service

Troubleshooting and Limitations

This section describes troubleshooting and limitations for JBoss discovery.

- **Limitation:** DFM can discover a J2EE application only when its .ear file is unzipped to a folder.
- **Limitation:** When using JBoss 7.x, this discovery only supports local Host Controller configuration, because JMX MBeans of such a managed JBoss server has no information about the remote Domain Controller.

Chapter 60: WebLogic Discovery

This chapter includes:

Overview

WebLogic discovery enables you to discover a full topology including J2EE applications, and JDBC and JMS resources.

Supported Versions

The following versions are supported:

WebLogic 6.x, 7.x, 8.x, 9.x, and 10.x, 11g, 11gR1 PS1, 11gR1 PS2, and 11gR1 PS3.

How to Discover WebLogic Topology by JMX

This task describes how to discover WebLogic. The WebLogic discovery process enables you to discover a complete WebLogic topology including J2EE applications, JDBC, and JMS resources.

DFM first finds WebLogic servers based on the JMX protocol, then discovers the WebLogic J2EE environment and components.

This task includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisite - Set up drivers" below
- "Run the discovery" on page 805

1. **Prerequisite - Set up protocol credentials**

   This discovery is based on the JMX protocol using credentials from the Weblogic protocol. Weblogic protocol credentials must be defined.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Set up drivers**

   Set up the drivers needed to discover WebLogic. Default WebLogic drivers are not included and should be copied to the Probe.

a. To discover WebLogic on SSL, obtain the following drivers:

Driver	Description
wlcipher.jar	If WebLogic is running on SSL **Note:** For all supported WebLogic versions
client trust store JKS file	If WebLogic is running on SSL. For example, **DemoTrust.jks**
jsafeFIPS.jar	If WebLogic is running on SSL **Note:** For WebLogic 8.1 SP5 and later
wlfullclient.jar	If WebLogic is running on SSL. **wlfullclient.jar** should be generated first using JarBuilder tool i. Change directory to %weblogic.home%/server/lib2. ii. Run java -jar wljarbuilder.jar **Note:** For WebLogic 9.x and 10.x
weblogic.jar	For WebLogic 8.x only
wlclient.jar	For WebLogic 9.x and 10.x only
wljmxclient.jar	For WebLogic 9.x and 10.x only

b. Place the drivers under the correct version folder in the following location:

```
C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
\discoveryResources\j2ee\weblogic\<version_folder>
```

For example,

```
C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
\discoveryResources\j2ee\weblogic\8.x
```

c. Restart the Probe before running the DFM jobs.

3. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **JEE TCP Ports** job to discover service endpoint information. For job details, see "JEE TCP Ports Job" on the next page.

   c. Run the **JEE Weblogic Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "JEE Weblogic Connections by JMX Job" on page 810.

   d. Run the **JEE Weblogic by JMX** job to perform a deep discovery of application server topology. For job details, see "JEE Weblogic by JMX Job" on page 813.

How to Discover WebLogic Topology by Shell

The WebLogic discovery process enables you to discover a complete WebLogic topology including J2EE applications, JDBC, and JMS resources. DFM first finds application servers based on the Shell protocol or endpoints (TCP Ports) and then discovers the WebLogic J2EE environment and components by shell.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the Shell protocol. Define credentials for one of the following protocols:

   - NTCMD protocol
   - SSH protocol
   - Telnet protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Discovery Workflow**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

a. Run the **Range IPs by ICMP** job to discover the target IPs.

b. Run the **Host Connection by Shell** job to discover the target host and shell connectivity to it.

c. Run one of the two jobs:

   - **Host Applications by Shell** to discover resources of the target host, including running processes.

   - **JEE TCP Ports** to discover service endpoint information. For job details, see "JEE TCP Ports Job" below.

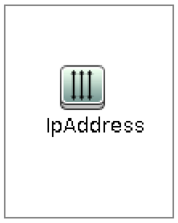
d. Run the job **JEE Weblogic by Shell**. For job details, see "JEE Weblogic by Shell Job" on page 818.

JEE TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host's ports.
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).

Name	Default Value	Description
ports	**For JEE TCP Ports job:** • weblogic, weblogicSSL, websphere_jmx, rmi **For Database TCP Ports job:** • oracle, db2, sybase, sql, mysql **For SAP TCP Ports job:** • sap, sap_jmx, sap_http, sap_https **For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.

Name	Default Value	Description
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

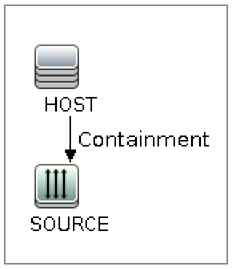
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

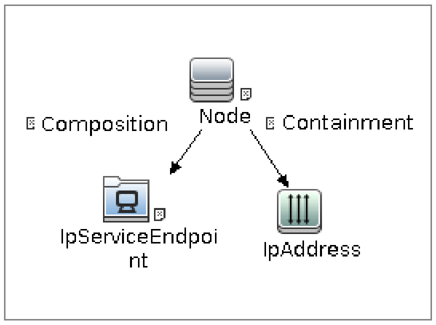
Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

JEE Weblogic Connections by JMX Job

This section includes details about the job.

Trigger Query



- Node Conditions

Node Name	Condition
Node	None
IpServiceEndPoint	IpServiceName Equal "weblogic"
IpAddress	NOT IP Probe Name Is null

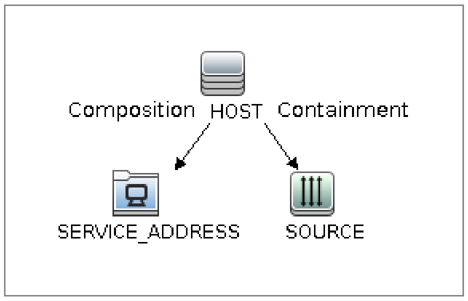
Job Parameters

Parameters are not overridden by default and use values from the adapter.

Adapter - JMX_J2EE_WebLogic_Connection

This adapter is used for Weblogic Server discovery.

- Input CIT: IpAddress
- Input Query:



- Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}
ports	${SERVICE_ADDRESS.network_port_number:NA}
hostId`	${HOST.root_id}

- Used Scripts

- connection.py
- db.py
- db_builder.py
- db_platform.py
- entity.py
- iteratortools.py
- j2eeutils.py
- jdbc.py
- jdbc_ulr_parser.py
- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- JMX_J2EE_WebLogic_Connection.py
- protocol.py
- weblogic.py
- weblogic_discoverer.py

- Global Configuration File: None
- Adapter Parameters

Name	Value	Description
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process.
runInSeparateProcess	true	Should pattern run in separate thread.

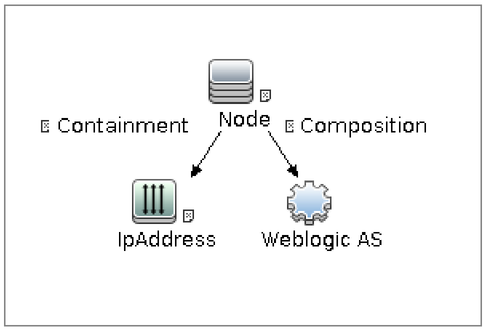
Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndPoint
- J2EE Domain
- JEE Node
- JVM
- Membership
- Node
- Usage
- WebLogic AS

JEE Weblogic by JMX Job

This section includes details about the job:

Trigger Query



- Node Conditions

Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
Weblogic AS	NOT Reference to the credentials dictionary entry Is null

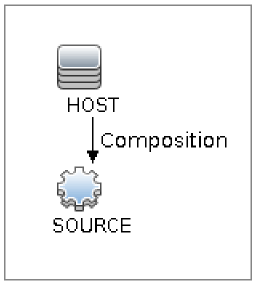
Job Parameters

Parameters are not overridden by default and use values from the adapter.

Adapter - JMX_J2EE_WebLogic

This adapter is used for Weblogic J2EE Topology Discovery by JMX.

- Input CIT: Weblogic AS
- Input Query:



- Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
port	${SOURCE.application_port}
servername	${SOURCE.name}
version	${SOURCE.application_version}
protocol	${SOURCE.j2eeserver_protocol}

- Used Scripts
  - connection.py
  - db.py
  - db_builder.py
  - db_platform.py
  - entity.py
  - iteratortools.py
  - j2eeutils.py
  - jdbc.py
  - jdbc_url_parser.py
  - jdbcutils.py
  - jee.py
  - jee_connection.py
  - jee_discoverer.py
  - jms.py
  - jmx.py

- JMX_J2EE_WebLogic.py
- protocol.py
- weblogic.py
- weblogic_discoverer.py

- Global Configuration File: globalSettings.xml
- Adapter Parameters

Name	**Value**	**Description**
deploymentDescriptors	true	Set to **true** to fetch deployment descriptors of J2EE Application, EJB Modules and Web Modules (value: true/false).
discoverAppResources	true	Discover modules, ejbs and servlets if set to true.
discoverJMSResources	true	Discover jms providers and jms servers if set to true.
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process.
runInSeparateProcess	true	Should pattern run in separate thread.
discoverDeployedOnly Applications	true	Discover applications that are deployed and are in running status

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database
- Dependency

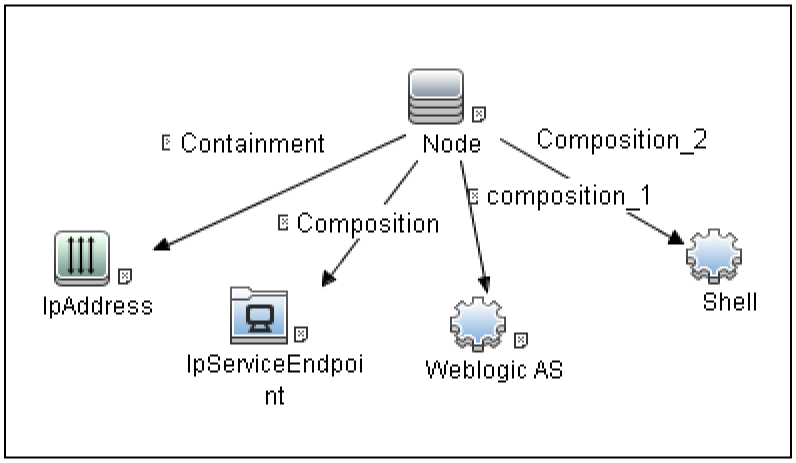
- Deployed
- IpAddress
- IpServiceEndPoint
- J2EE Cluster
- J2EE Domain
- J2EE Execute Queue
- J2EE Managed Object
- JDBC Data Source
- JEE Node
- Membership
- Node
- Usage
- Web Service
- Weblogic AS

Note: JDBC Datasources cannot be discovered if they were not activated in the Weblogic Admin Console prior to discovery.

JEE Weblogic by Shell Job

This section includes details about the job:

Trigger Query



Node Name	Condition
Node	none
Shell	NOT Reference to the credentials dictionary entry Is null
Weblogic AS	None
IpAddress	NOT IP Probe Name Is null
IpServiceEndPoint	IpServiceName Equal weblogic OR IpServiceName Equal weblogicSSL OR ServiceNames Contains weblogic OR ServiceNames Contains weblogicSSL

Parameters

Parameters are not overridden by default and use values from the adapter.

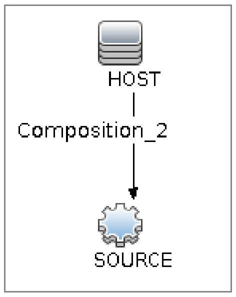
Adapter Information

This job uses the **WebLogic_By_Shell** adapter.

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}
Protocol	${SOURCE.root_class}

Used Scripts

- connection.py
- db.py
- db_builder.py
- db_platform.py
- entity.py
- file_ver_lib.py
- fptools.py
- iteratortools.py
- jdbc.py

- jdbc_url_parser.py
- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- process.py
- process_discoverer.py
- protocol.py
- weblogic.py
- weblogic_by_shell.py
- weblogic_discoverer.py

Global Configuration File

globalSettings.xml

Adapter Parameters

None

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database

- DatabaseSchema
- Dependency
- Deployed
- IpAddress
- IpServiceEndPoint
- J2EE Cluster
- J2EE Domain
- J2EE Managed Object
- JDBC Data Source
- JEE Node
- Membership
- Node
- Usage
- Web Service
- Weblogic AS

Troubleshooting and Limitations

Troubleshooting

- **Problem:** When running the WebLogic by JMX job, using the SSL protocol, and the UCMDB server and Data Flow Probe are connected using the SSL protocol, the job is unable to connect to the target node.

  The following are alternative solutions:

  Solution 1: Configure an HTTP connection between UCMDB server and the Data Flow Probe.

Solution 2: Allow a non SSL connection to the WebLogic server and configure UCMDB JMX credentials; do not use an SSL connection

Solution 3: Update the parameter **remoteJVMArgs** of the jobs (JEE WebLogic Connections by JMX job and JEE WebLogic by JMX job) by adding the following argument:

```
Djavax.net.ssl.trustStore=..\runtime\probeManager\discoveryResources
\j2ee\websphere\UCMDB_store.jks
```

Limitations

- For Weblogic versions 8.x and earlier, DFM discovers only those domains created by the WebLogic Configuration Wizard.
- For versions earlier than WebLogic 9, the JEE WebLogic by Shell job can run only on admin server hosts. For WebLogic version 9 or later, the job can run also on hosts that contain managed nodes only.
- DFM can discover a J2EE application only when its .ear file is unzipped to a folder.
- The WebLogic installation includes an example that is filtered out by default. You can remove the filter in the **weblogic_by_shell.py** Jython script. Look for **WL_EXAMPLE_DOMAINS = 'medrec'**.

Chapter 61: WebSphere Discovery

This chapter includes:

Overview

This section describes how to discover WebSphere application center. The WebSphere discovery process enables you to discover the complete WebSphere topology including J2EE applications, JDBC, and JMS resources.

Supported Versions

WAS Version	**J2EE Version**	**JVM Version**
5.0	J2EE 1.3	JVM 1.3
5.1	J2EE 1.3	JVM 1.4
6.0	J2EE 1.4	JVM 1.4
6.1	J2EE 1.4	JVM 1.5
7.0	Java EE 5	JVM 1.6

How to Discover WebSphere Topology by JMX

DFM first finds WebSphere servers based on either SOAP or RMI authentication, then discovers the WebSphere J2EE environment and components.

This task describes how to discover WebSphere connections by JMX, and includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery is based on the JMX protocol using credentials from the WebSphere protocol. WebSphere protocol credentials must be defined.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Set up drivers**

   Set up the drivers needed to discover WebSphere. Default WebSphere drivers are included by default with the Probe installation.

   The Probe installation includes WebSphere drivers for versions 5 and 6, but you can use your own drivers, if you prefer. However, you can use only drivers that work with a supported version. For details on supported versions, see Discovered Applications.

   To update the .jar files:

   a. Copy the drivers to the correct version folder in the following location:

      ```
      C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
      \discoveryResources\j2ee\websphere\<version_folder>
      ```

      For example,

      ```
      C:\hp\UCMDB\DataFlowProbe\runtime\probeManager
      \discoveryResources\j2ee\websphere\5.x
      ```

   b. Restart the Probe before running the DFM jobs.

3. **Update the .jar files**

   a. Copy the following files from a Websphere 8.x application server:

- **ibmorb.jar** from the following location: **<WebSphere root folder>**\AppServer\java\jre\lib\ibmorb.jar
- All files from the following location: **<WebSphere root folder>**\AppServer\deploytool\itp\plugins\<com.ibm.websphere.v8_....>\wasJars\

  where **<WebSphere root folder>** is the folder where you installed WebSphere.

b. Stop the Data Flow Probe.

c. Backup all of the files in **<DataFlowProbe root folder>**\runtime\probeManager\discoveryResources\j2ee\websphere\ except **UCMDB_store.jks**.

   where **<DataFlowProbe root folder>** is the folder where you installed the Data Flow Probe.

d. Delete all files from **<DataFlowProbe root folder>**\runtime\probeManager\discoveryResources\j2ee\websphere\ except UCMDB_store.jks.

e. Put all of the files you copied in step a in the following location on the Data Flow Probe: **<DataFlowProbe root folder>**\runtime\probeManager\discoveryResources\j2ee\websphere

f. Restart the Data Flow Probe.

4. Run the discovery

Run the following jobs in the following order:

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

a. Run the **Range IPs by ICMP** job to discover the target IPs.

b. Run the **JEE TCP Ports** job to discover service endpoint information. For job details, see "JEE TCP Ports Job" on page 829.

c. Run the **JEE WebSphere Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "JEE WebSphere Connections by JMX Job" on page 833.

d. Run the **JEE WebSphere by JMX** job to perform a deep discovery of application server topology. For job details, see "JEE WebSphere by Shell or JMX Job" on page 837.

How to Discover WebSphere Topology by Shell

This task describes how to discover a complete WebSphere topology using Shell protocols. The WebSphere discovery process discovers Web services that are deployed on an IBM WebSphere server. The discovered Web services are represented by the `webservice` CIT in the CMDB.

DFM first finds application servers based on the Shell protocol or endpoints (TCP Ports) and then discovers the WebSphere J2EE environment and components by Shell.

This task includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisite - Set up key stores" below
- "Run the discovery" on the next page

1. Prerequisite - Set up protocol credentials

This discovery uses the Shell protocol. You must define one of the following protocols:

- SSH protocol
- Telnet protocol
- NTCMD protocol

For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. Prerequisite - Set up key stores

The following procedure is relevant if you are running a client machine that includes two key stores, each one needed for identification on a specific WebSphere server. If the client attempts to connect to one of the WebSphere servers with the wrong key store, the attempt fails. If the client then uses the second, correct key store to connect to the WebSphere server, that attempt also fails.

- **Solution 1**: Set up one key store on the client for all WebSphere servers.
- **Solution 2**: Set up one key store per IP address range for all WebSphere servers that use the

same user name and password. For a server that uses a different user name and password, set up a key store in another IP range.

3. **Run the discovery**

   Run the following jobs in the following order:

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **Host Connection by Shell** job to discovers the target host and Shell connectivity to the host.

   c. Run one of the following jobs:

      - Run the **Host Applications by Shell** job to discover applications of the target host, including running processes.

      - Run the **JEE TCP Ports** job to discover service endpoint information. For job details, see "JEE TCP Ports Job" on the next page.

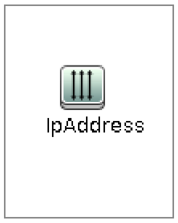
   d. Run the **JEE WebSphere by Shell** job. For job details, see "JEE WebSphere by Shell Job" on page 842.

JEE TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host's ports.
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).

Name	Default Value	Description
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).
ports	**For JEE TCP Ports job:** • weblogic, weblogicSSL, websphere_jmx, rmi **For Database TCP Ports job:** • oracle, db2, sybase, sql, mysql **For SAP TCP Ports job:** • sap, sap_jmx, sap_http, sap_https **For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).

Name	Default Value	Description
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

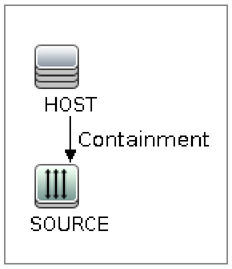
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

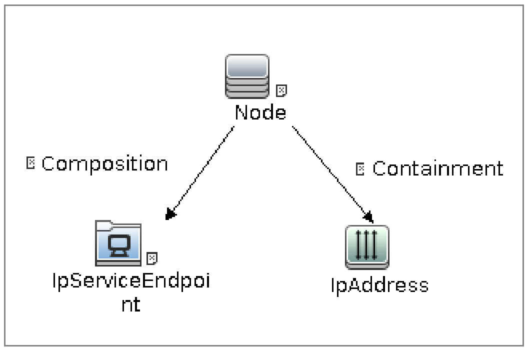
Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

JEE WebSphere Connections by JMX Job

This section includes details about the job.

Trigger Query



Node Conditions

Node Name	Condition
Node	None
IpServiceEndPoint	IpServiceName Equal "websphere"
IpAddress	NOT IP Probe Name Is null

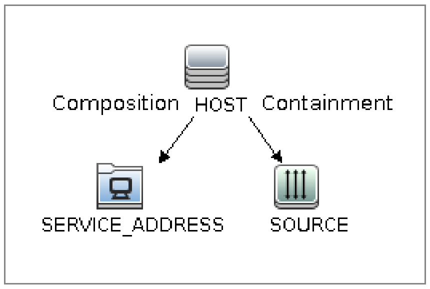
Job Parameters

Parameters are not overridden by default and use values from the adapter.

Adapter - JMX_J2EE_WebSphere_Connection

This adapter is used for WebSphere Server discovery.

- Input CIT: IpAddress
- Input Query:



- Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}
ports	${SERVICE_ADDRESS.network_port_number:NA}
hostId	${HOST.root_id}
ip_dnsname	${SOURCE.authoritative_dns_name:NA}

- Used Scripts
  - connection.py
  - db.py
  - db_builder.py
  - db_platform.py
  - entity.py
  - iteratortools.py
  - j2eeutils.py
  - jdbc.py
  - jdbc_url_parser.py
  - jdbcutils.py

- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- JMX_J2EE_WebSphere_Connection.py
- protocol.py
- websphere.py

- Global Configuration File: None
- Parameters

Name	Value	Description
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process.
runInSeparateProcess	true	Should pattern run in separate thread.

Discovered CITs

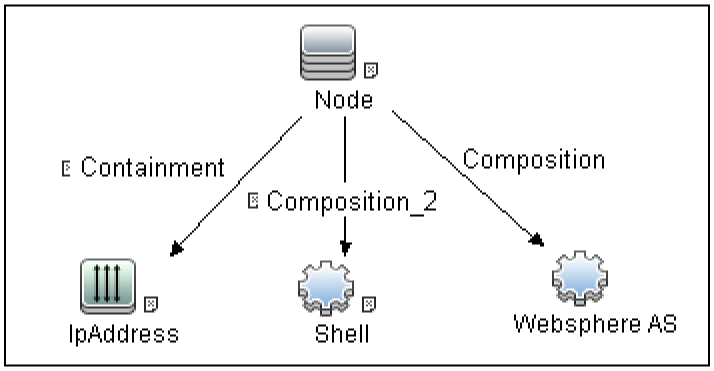
- Composition
- IpAddress
- IpServiceEndPoint
- J2EE Domain
- JEE Node
- Node

- Usage
- Websphere AS

JEE WebSphere by Shell or JMX Job

This section includes details about :

Trigger Query



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
Websphere AS	NOT Reference to the credentials dictionary entry Is null
Shell	NOT Reference to the credentials dictionary entry Is null

Job Parameters

Parameters are not overridden by default and use values from the adapter.

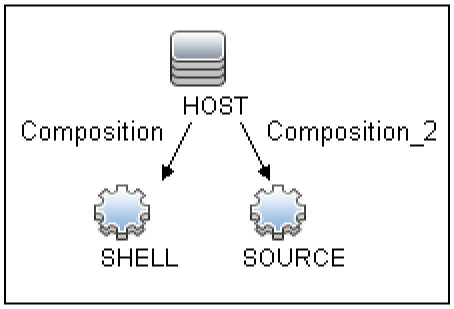
Adapter Information

This job uses the **JMX_J2EE_WebSphere** adapter.

Input CIT

WebSphere AS

Input Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip	${SHELL.application_ip:NA}
ip_address	${SHELL.application_ip}
port	${SHELL.application_port:NA}
protocol	${SHELL.root_class:NA}
shellCredentialsId	${SHELL.credentials_id:NA}
version	${SOURCE.application_version}

Used Scripts

- connection.py
- core.py
- db.py
- db_builder.py
- db_platform.py
- entity.py

- iteratortools.py
- j2eeutils.py
- jdbc.py
- jdbc_url_parser.py
- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- JMX_J2EE_WebSphere.py
- protocol.py
- websphere.py
- websphere_discoverer.py

Global Configuration File

globalSettings.xml

Adapter Parameters

Name	Value	Description
applications	None	List of applications to discover (comma separated).
discoverAppResources	true	Discover modules, ejbs and servlets if set to true.
discoverConfigFile	true	Discover additional configuration files for cell, server, and application, if set to true.

Name	Value	Description
discoverEAR	true	Discover J2EE application EAR files if set to true.
discoverJDBCResources	true	Discover jdbc providers and datasources if set to true.
discoverJMSResources	true	Discover jms providers and jms servers if set to true.
remoteJVMArgs	-Xms64m -Xmx256m -XX:PermSize=256m -XX:MaxPermSize=256m	JVM parameters that should be passed to the remote process.
runInSeparateProcess	true	Determines if the pattern should be run in a separate thread.
servers	None	List of servers to discover (comma separated).

Discovered CITs

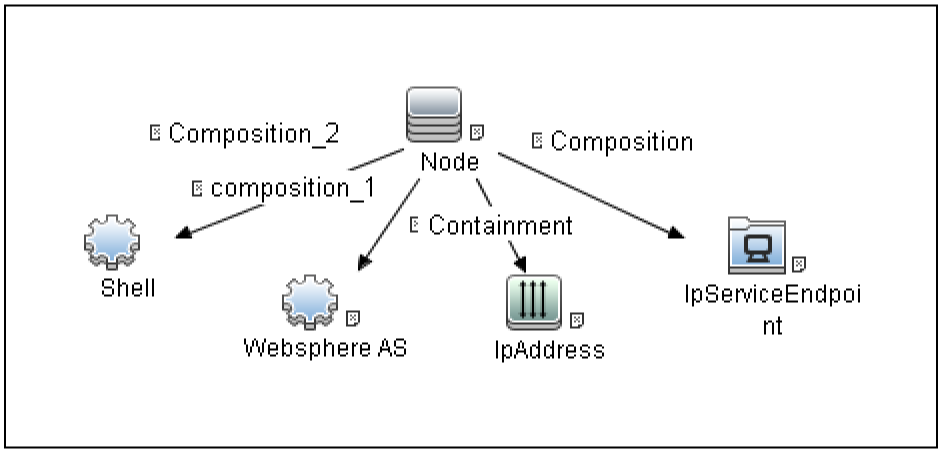
- Composition
- ConfigurationDocument
- Database
- Dependency
- Deployed
- IpAddress
- IpServiceEndPoint
- J2EE Cluster
- J2EE Domain
- J2EE Managed Object
- JDBC Data Source

- JEE Node
- Membership
- Node
- Usage
- Web Service
- Websphere AS

JEE WebSphere by Shell Job

This section includes details about the job.

Trigger Query



Node Name	Condition
Node	None
IpAddress	NOT IP Probe Name Is null
Websphere AS	NOT Reference to the credentials dictionary entry Is null
Shell	None
IpServiceEndPoint	IpServiceName Equal websphere_jmx OR ServiceNames Contains websphere_jmx

Job Parameters

Parameters are not overridden by default and use values from the adapter.

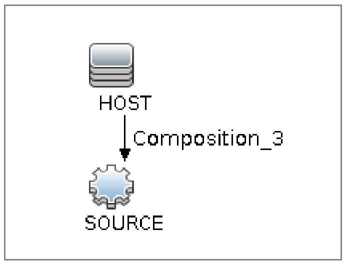
Adapter Information

This job uses the **WebSphere_By_Shell** adapter.

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}

Used Scripts

- connection.py
- db.py
- db_builder.py
- db_platform.py
- entity.py
- file_ver_lib.py
- fptools.py
- iteratortools.py
- jdbc.py
- jdbc_url_parser.py

- jdbcutils.py
- jee.py
- jee_connection.py
- jee_discoverer.py
- jms.py
- jmx.py
- process.py
- process_discoverer.py
- protocol.py
- websphere.py
- websphere_by_shell.py
- websphere_discoverer.py

Global Configuration File

globalSettings.xml

Adapter Parameters

None

Discovered Elements

DFM discovers the following elements:

- The Version Number

  DFM discovers the version number of the WebSphere application server from the **WAS.product** or **BASE.product** file (depending on the WebSphere version) in the **<WebSphere base directory>\properties\version** folder.

- The Server Listening Port and Address

DFM retrieves information about WebSphere servers by searching for the **serverindex.xml** file, found either in the **<WebSphere base directory>\profiles\<PROFILE>\config\cells\<CELL>\nodes\<NODE>** folder, or the **<WebSphere base directory>\config\cells\<CELL>\nodes\<NODE>** folder.

- J2EE Applications

  DFM searches for the **deployment.xml** file in each **<WebSphere base directory>\profiles\<PROFILE>\config\cells\<CELL>\applications** folder (or in the **<WebSphere base directory>\config\cells\<CELL>\nodes\<NODE>\applications** folder). The **deployment.xml** file is located in every installed application folder, and contains information about application targets.

- Configuration Files

  DFM creates CIs for the **resources.xml** resources configuration file. A CI is created for each cell, node, and server (with the relevant prefix); each CI is attached to the WebSphere server CI.

- JMS Resources

  Websphere JMS resources are configured as JMS providers. Resources are of two main kinds: **connection factories** and **destinations** (topic, queue). These may be further categorized as follows:

  - Connection Factories
    - resources.jms.mqseries:MQConnectionFactory
  - Queue Connection Factories
    - resources.jms.mqseries:MQQueueConnectionFactory
    - resources.jms.internalmessaging:WASQueueConnectionFactory
  - Topic Connection Factories
    - resources.jms.mqseries:MQTopicConnectionFactory
    - resources.jms.internalmessaging:WASTopicConnectionFactory
  - Queues or Topics
    - resources.jms:GenericJMSDestination
    - resources.jms.mqseries:MQTopic

- resources.jms.mqseries:MQQueue
- resources.jms.internalmessaging:WASTopic
- resources.jms.internalmessaging:WASQueue

DFM strives to use all the types mentioned to acquire information about used resources. Discovery looks for the configuration file **resources.xml** on different deployment scopes. The following table shows the deployment scopes and relative path to the configuration file.

Scope	Relative File Path
Cell	<PROFILE>\config\cells\<CELL>\resources.xml
Cluster	<PROFILE>\config\cells\<CELL>\clusters\<CLUSTER>\resources.xml
Node	<PROFILE>\config\cells\<CELL>\nodes\<NODE>\resources.xml
Server	<PROFILE>\config\cells\<CELL>\nodes\<NODE>\servers\<SERVER>\resources.xml

Note: The file path is relative to the <PROFILE> home directory.

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Database
- Database Schema
- Dependency
- Deployed
- IpAddress
- IpServiceEndPoint

- J2EE Cluster
- J2EE Domain
- J2EE Managed Object
- JDBC Data Source
- JEE Node
- Membership
- Node
- Usage
- Web Service
- Websphere AS

Troubleshooting and Limitations

This section describes troubleshooting and limitations for WebSphere discovery.

Troubleshooting

- **Problem:** When running the Websphere by JMX job, using the SSL protocol, and the UCMDB server and Data Flow Probe are connected using the SSL protocol, the job is unable to connect to the target node.

  The following are alternative solutions:

  Solution 1: Configure an HTTP connection between UCMDB server and the Data Flow Probe.

  Solution 2: Allow a non SSL connection to the Websphere server and configure UCMDB JMX credentials; do not use an SSL connection

  Solution 3: Update the parameter **remoteJVMArgs** of the jobs (JEE WebSphere Connections by JMX job and JEE WebSphere by Shell or JMX job) by adding the following argument:

  ```
  Djavax.net.ssl.trustStore=..\runtime\probeManager\discoveryResources
  \j2ee\websphere\UCMDB_store.jks
  ```

Limitations

- If DFM finds two cells with the same name on the same host, only one cell configuration (**j2eedomain** topology) is reported.
- EJB and Web Service CIs are not discovered.
- DFM can discover a J2EE application only when its **.ear** file is unzipped to a folder.
- A job (script) works with a certificate in jks* key format only.

Part 11: Middleware > Messaging Servers

Chapter 62: Microsoft MQ (Message Queue) Discovery

This chapter includes:

Supported Versions

MS-MQ discovery supports MS MQ version 3.0 or later.

How to Discover Microsoft MQ

The Microsoft Message Queue (MS MQ) discovery process enables you to discover MS MQ topology running with Active Directory, as well as the end configuration of all MS MQ servers.

There are two discovery flows, detailed as follows:

1. **Run the discovery by LDAP**

   a. Run the **Range IPs by ICMP** job, or the **Range IPs by nmap** job, to discover the MS MQ system IP addresses.

   b. Run the **TCP Ports** job to discover the LDAP ports on the MS MQ system.

   c. Run the **Active Directory Connection by LDAP** job to detect which LDAP credentials are needed for discovery for the **Microsoft Message Queue Topology by LDAP** job.

   d. Run the **Microsoft Message Queue Topology by LDAP** job to discover the Active Directory topology (forest, site-link).

2. **Run the discovery by NTCMD or UDA**

   a. Run the **Range IPs by ICMP** job, or the **Range IPs by nmap** job to to discover the MS MQ system IP addresses.

   b. Run the **Host Connection by Shell** job to detect which Shell credentials are needed for discovery for the **Host Applications by Shell** job.

   c. Run the **Host Applications by Shell** job. At this stage, UCMDB contains information about the MS MQ Manager and machine with the domain controller, on condition that the server (the physical machine on which the MS MQ is installed) is a member of the domain.

   d. Run the **Microsoft Message Queue Topology by NTCMD or UDA** job to discover the server side topology (queues, triggers, rules).

Note: Because information is retrieved from configuration files in three short registry branches only, and each file is less than 2 KB, system performance should not be affected.

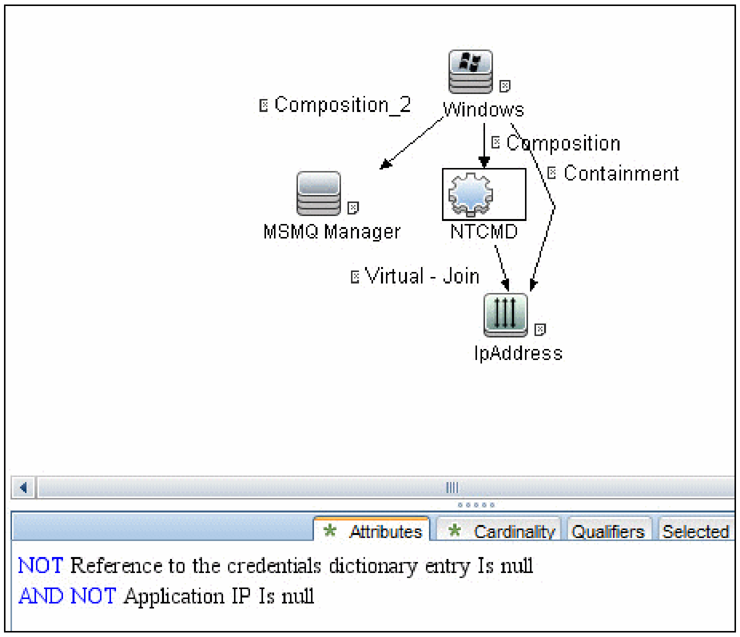
For details on how DFM discovers MQ topology, see "Microsoft MQ Topology Discovery Methodology" on page 858.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Microsoft Message Queue Topology by NTCMD or UDA Job

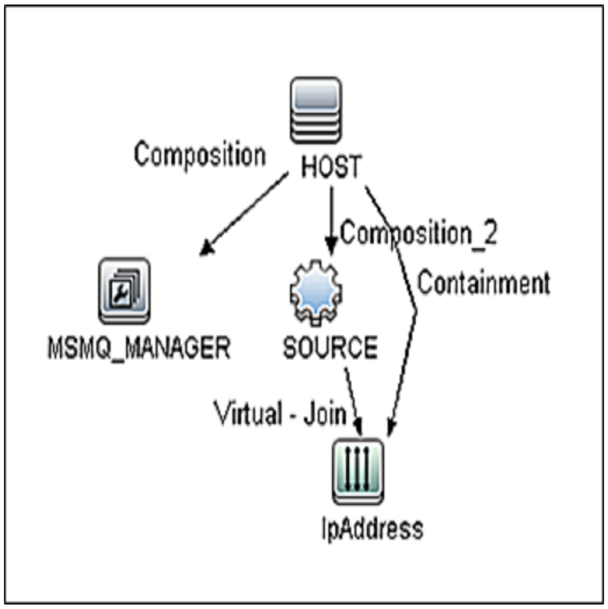
This section includes details about the job.

Trigger Query



NOT Reference to the credentials dictionary entry Is null
AND NOT Application IP Is null

Input Query



Node Name	Condition
SOURCE	CI Type Equal ntcmd OR CI Type equal uda
HOST	None
MSMQ_MANAGER	None
IpAddress	NOT IP Probe Name Is Null

Discovered CITs

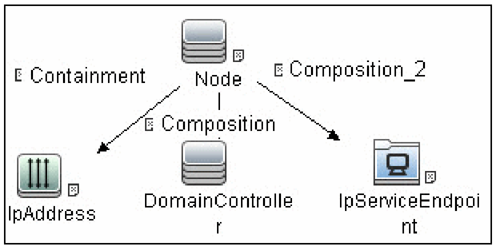
- Composition
- Containment
- IpAddress
- MSMQ Manager
- MSMQ Queue
- MSMQ Rule

- MSMQ Trigger
- Node
- Usage

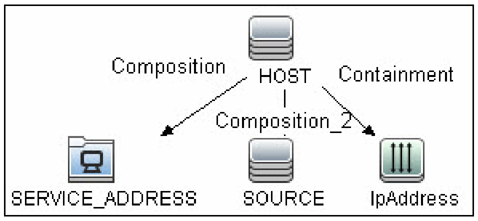
Microsoft Message Queue Topology by LDAP Job

This section includes details about the job.

Trigger Query



Input Query



Discovered CITs

- Active Directory Forest
- ActiveDirectorySite
- ActiveDirectorySystem
- Composition
- Containment
- IpAddress
- MSMQ Manager
- Membership
- Node

- Usage

Adapter Parameters

- **baseDn.** This value determines the DN under which records about domain controller servers are stored. The default value is **OU=Domain Controllers**.

Microsoft MQ Discovery Scripts

To view the scripts, go to **Adapter Management > Discovery Packages > Microsoft_MQ > Scripts**.

Script	Description
ntcmd_msmq.py	Main script for the **Microsoft Message Queue Topology by NTCMD or UDA** job
ldap_msmq.py	Main script for the **Microsoft Message Queue Topology by LDAP** job
plugin_microsoft_ mq.py	Shallow plug-in for MS MQ Manager discovery (**Adapter Management > Discovery Packages > Host_Resources_Basic > Scripts**)
host_resolve_utils.py	DNS resolving utilities **(Adapter Management > Discovery Packages > Host_Resources_Basic > Scripts**)

Microsoft MQ Topology Discovery Methodology

This section describes how DFM discovers the MS MQ topology.

This section includes the following topics:

Host Applications by Shell Job

This job uses the **plugin_microsoft_mq.py** script.

Information is parsed from the following registry branches:

Registry Branch (1)

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters\MachineCache\

- **Command Output**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters\MachineCache
EnterpriseId     REG_BINARY     C209A2FE9203F64CB543441CC92A40DC
    SiteId     REG_BINARY     FB7BA54DFF5F40429ECA64752D0130A0
    MQS_DepClients     REG_DWORD     0x0
    MQS     REG_DWORD     0x1
    MQS_DsServer     REG_DWORD     0x0
    MQS_Routing     REG_DWORD     0x1
    QMId     REG_BINARY     1D19B008D7BF654B84050FC7353F993C
    MachineQuota     REG_DWORD     0x100000
    MachineJournalQuota     REG_DWORD     0xffffffff
    LongLiveTime     REG_DWORD     0x54600
```

- **Regular Expression Patterns**

  Message routing enabled:

  ```
  "\s*MQS_Routing\s+REG_DWORD\s+0x[0]*(\d)\s*"
  ```

  Message storage limit:

```
"\s*MachineQuota\s+REG_DWORD\s+(\w+)\s*"
```

Message journal limit:

```
"\s*MachineJournalQuota\s+REG_DWORD\s+(\w+)\s*"
```

Registry Branch (2)

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters\setup\`

- **Command Output**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters\setup
MachineDomain    REG_SZ    UCMDB-EX
MachineDomainFQDN    REG_SZ    ucmdb-ex.dot
OSType    REG_DWORD    0x500
CreateMsmqObj    REG_DWORD    0x0
UserSid    REG_BINARY 10500000000000515000000576A62162631895
C45612C98F4010000
MachineDN    REG_SZ    CN=MSMQ-VM01,CN=Computers,DC=ucmdb-ex,DC=dot
JoinStatus    REG_DWORD    0x2
MSMQAddedToICFExceptionList    REG_DWORD    0x1
MQDSSvcInstalled    REG_DWORD    0x1
InetpubWebDir    REG_DWORD    0x1
```

- **Regular Expression Patterns**

Machine domain name:

```
 "\s*MachineDomainFQDN\s+REG_SZ\s+([\w\-\.]+)\s*"
```

Registry Branch (3)

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Setup\

- **Command Output**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Setup
    msmq_Core    REG_DWORD    0x1
    msmq_LocalStorage    REG_DWORD    0x1
    msmq_ADIntegrated    REG_DWORD    0x1
    InstalledComponents    REG_DWORD    0xf8000000
    msmq_MQDSService    REG_DWORD    0x1
    msmq_TriggersService    REG_DWORD    0x1
    msmq_HTTPSupport    REG_DWORD    0x1
    msmq_RoutingSupport    REG_DWORD    0x1
```

- **Regular Expression Patterns**

  MsMQ is a domain member:

  ```
  "\s*msmq_ADIntegrated\s+REG_DWORD\s+0x[0]*(\d)\s*"
  ```

  Triggers enabled:

  ```
  "\s*msmq_TriggersService\s+REG_DWORD\s+0x[0]*(\d)\s*"
  ```

Microsoft Message Queue Topology by NTCMD or UDA Job

This job discovers the settings and relationships of triggers, rules, and queues.

MS MQ Queue Discovery

- **Registry Branch**

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters /v StoreReliablePath
  ```

- **Command Output**

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters
      StoreReliablePath    REG_SZ    C:\WINDOWS\system32\msmq\storage
  ```

- **Regular Expression Patterns**

  Base parent folder for message storage

  ```
  "\s*StoreReliablePath\s+REG_SZ\s+(.+)"
  ```

- **Command**

  ```
  dir /B /A:-D <ms mq queue settings folder>
  ```

- **Command Output**

  ```
  dir /B /A:-D C:\WINDOWS\system32\msmq\storage\lqs
  00000002.990736e8
  00000003.6ab7c4b8
  00000004.4c1eb11b
  00000006.e2f46f06
  00000010.d1c14377
  00000012.e6d243aa
  9b0b035bf61b429d845bbd61740403b7.0d0d6ec1
  ```

- **Result**

  The file names of MS MQ queue configurations are retrieved. DFM then iterates against this list of files, reads them, and parses the queue settings.

- **Command**

  ```
  type <full_path_to_the_file>
  ```

- **Command Output**

  ```
  type C:\WINDOWS\system32\msmq\storage\lqs\00000002.990736e8
  [Properties]
  Label=private$\admin_queue$
  Type=00000000-0000-0000-0000-000000000000
  QueueName=\private$\admin_queue$
  Journal=00
  Quota=4294967295
  Security=010007805c00000068000000000000001400000000200
  48000300000000018003f000e00010200000000000520000000020
  0200000000140024000200010100000000000100000000000140
  0040000000101000000000005070000000101000000000000051200
  0000010100000000000512000000
  JournalQuota=4294967295
  CreateTime=1259681363
  BasePriority=32767
  ModifyTime=1259681363
  Authenticate=00
  PrivLevel=1
  Transaction=00
  SystemQueue=01
  Signature=DoronJ
  ```

- **Parse Rules**

  Queue name:

  `".*QueueName\s*=\s*(.+?)\n.*"`

  Is transactional:

  `".*Transaction\s*=\s*(\d+).*"`

  Queue type (public/private):

  `"^[\\]*(private).*$" against Queue name`

  Message limit:

  `".*\s+Quota\s*=\s*(\d+).*"`

  Is journal enabled:

  `".*Journal\s*=\s*(\d+).*"`

  Journal limit:

```
".*JournalQuota\s*=\s*(\d+).*"
```

MS MQ Trigger Discovery

- **Registry Branch**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\Data\Triggers\
```

- **Command Output**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers
\Data\Triggers\31b8e2c4-f412-431e-9b2c-517f7e5031d7
    Name    REG_SZ    Test Trigger
    Queue    REG_SZ    msmq-vm2\Test Queue
    Enabled    REG_DWORD    0x1
    Serialized    REG_DWORD    0x0
    MsgProcessingType    REG_DWORD    0x1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\31b8e2c4-f412-431e-9b2c-517f7e5031d7\AttachedRules
    Rule0    REG_SZ    9c172d69-c832-453e-826b-4415b7d0dfef
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\728b0d45-531d-4887-9762-3191b0069bb1
    Name    REG_SZ    remote Trigger
    Queue    REG_SZ    msmq-vm01\Test Queue
    Enabled    REG_DWORD    0x1
    Serialized    REG_DWORD    0x0
    MsgProcessingType    REG_DWORD    0x0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\728b0d45-531d-4887-9762-3191b0069bb1\AttachedRules
    Rule0    REG_SZ    9c172d69-c832-453e-826b-4415b7d0dfef
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\b900d598-e3c2-4958-bf21-c8c99ed264e2
    Name    REG_SZ    qqqqqqq
    Queue    REG_SZ    msmq-vm2\private$\Private Test Queue
    Enabled    REG_DWORD    0x1
    Serialized    REG_DWORD    0x0
    MsgProcessingType    REG_DWORD    0x1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\b900d598-e3c2-4958-bf21-c8c99ed264e2\AttachedRules
    Rule0    REG_SZ    9c172d69-c832-453e-826b-4415b7d0dfef
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\dc4302f0-d28c-40e4-a19a-492dcee231fe
    Name    REG_SZ    Test2
    Queue    REG_SZ    msmq-vm2\private$\Test Transactional
    Enabled    REG_DWORD    0x1
    Serialized    REG_DWORD    0x1
    MsgProcessingType    REG_DWORD    0x2
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
Triggers\Data\Triggers\dc4302f0-d28c-40e4-a19a-492dcee231fe\AttachedRules
    Rule0    REG_SZ    9c172d69-c832-453e-826b-4415b7d0dfef
    Rule1    REG_SZ    2874c4c1-57f1-4672-bbdd-0c16f17788cf
```

MS MQ Rule Discovery

- **Regular Expression Patterns**

  The output buffer is split by the following regular expression:

  ```
  "(HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\
  Triggers\Data\Triggers\[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\
  -[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12})\s*\n"
  ```

  After each string buffer is split, the following patterns are applied:

  Trigger name:

  ```
  ".*Name\s+REG_SZ\s+(.*?)\n.*"
  ```

  Trigger GUID:

  ```
  " HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\
  Data\Triggers\([0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\
  -[0-9a-fA-F]{4}\-[0-9a-fA-F]{12})\s*\n"
  ```

  Assigned queue:

  ```
  ".*Queue\s+REG_SZ\s+(.*?)\n.*"
  ```

  Trigger is serialized:

  ```
  ".*Serialized\s+REG_DWORD\s+0x(\d+).*"
  ```

  Trigger is enabled:

  ```
  ".*Enabled\s+REG_DWORD\s+(0x\d+).*"
  ```

  Trigger message processing type:

  ```
  ".*MsgProcessingType\s+REG_DWORD\s+(0x\d+).*"
  ```

  Trigger assigned rule GUID:

```
".*Rule\d+\s+REG_SZ\s+([0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\
-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}).*"
```

- **Registry Branch**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\Data\Rules\
```

- **Command Output**

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\Data\Rules\
2874c4c1-57f1-4672-bbdd-0c16f17788cf
    Name    REG_SZ    Test Rule2
    Description    REG_SZ    bla bla
    ImplementationProgID    REG_SZ    MSQMTriggerObjects.MSMQRuleHandler
    Condition    REG_SZ    $MSG_PRIORITY_EQUALS=1
        $MSG_LABEL_DOES_NOT_CONTAIN=bla
    Action    REG_SZ    EXE     C:\WINDOWS\system32\calc.exe
    ShowWindow    REG_DWORD    0x1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\Data\Rules\
9c172d69-c832-453e-826b-4415b7d0dfef
    Name    REG_SZ    Test Rule
    Description    REG_SZ
    ImplementationProgID    REG_SZ    MSQMTriggerObjects.MSMQRuleHandler
    Condition    REG_SZ    $MSG_LABEL_CONTAINS=Test
    Action    REG_SZ    EXE     C:\WINDOWS\NOTEPAD.EXE
    ShowWindow    REG_DWORD    0x1
```

- **Regular Expression Patterns**

The output buffer is split by the following constant:

```
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Triggers\Data\Rules\"
```

After each string buffer is split, the following patterns are applied:

Rule name:

```
".*Name\s+REG_SZ\s+(.*?)\n.*"
```

Rule condition:

```
".*Condition\s+REG_SZ\s+(.*?)\n.*"
```

Rule action:

```
".*Action\s+REG_SZ\s+(.*?)\n.*"
```

Rule GUID:

```
"\s*([0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\
-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}).*"
```

Microsoft Message Queue Topology by LDAP Job

This job reports the Active Directory-related part of MS MQ deployment: AD Forest, AD Site, MS MQ Manager, and MS MQ Routing Link.

Schema parameters:

```
CN=Configuration,DC=<domain_name>,DC=<domain_suffix>
```

Site discovery (derived from AD discovery):

```
CN=Sites,CN=Configuration,<domain_name>,DC=<domain_suffix>
```

Server Discovery with MS MQ Manager

- **Branch**

  ```
  CN=Servers,CN=<site_name>,CN=Sites,CN=Configuration,DC=<domain_name>,DC=<domain_
  suffix>
  ```

- **Values**

  Server name property:

  ```
  'name'
  ```

  Server full DN:

  ```
  'distinguishedName'
  ```

  If an underlying branch exists (for `objectClass=mSMQSettings`), the server is considered to include an MS MQ Manager.

Chapter 63: TIBCO BusinessWorks and EMS Discovery

This chapter includes:

Overview

TIBCO Enterprise Message Service (**EMS**) is a messaging platform which combines different IT resources on a common enterprise backbone to manage real-time information flow.

TIBCO ActiveMatrix BusinessWorks (**BusinessWorks**) is a service creation, orchestration, and integration product, entirely created using open standards.

The TIBCO discovery process allows you to discover a full topology.

Discovery Mechanism

Because TIBCO does not have any system configuration files about applications, the TIBCO discovery mechanism starts by using TIBCO's **AppManage** utility to export a list of xml files to a temporary folder on the BusinessWorks server and by using TIBCO's **TibcoEmsAdmin** utility to get information about EMS and JMS topology.

The discovery mechanism continues with the **TIBCO BusinessWorks by Shell** and **TIBCO EMS by Shell** jobs.

Supported Versions

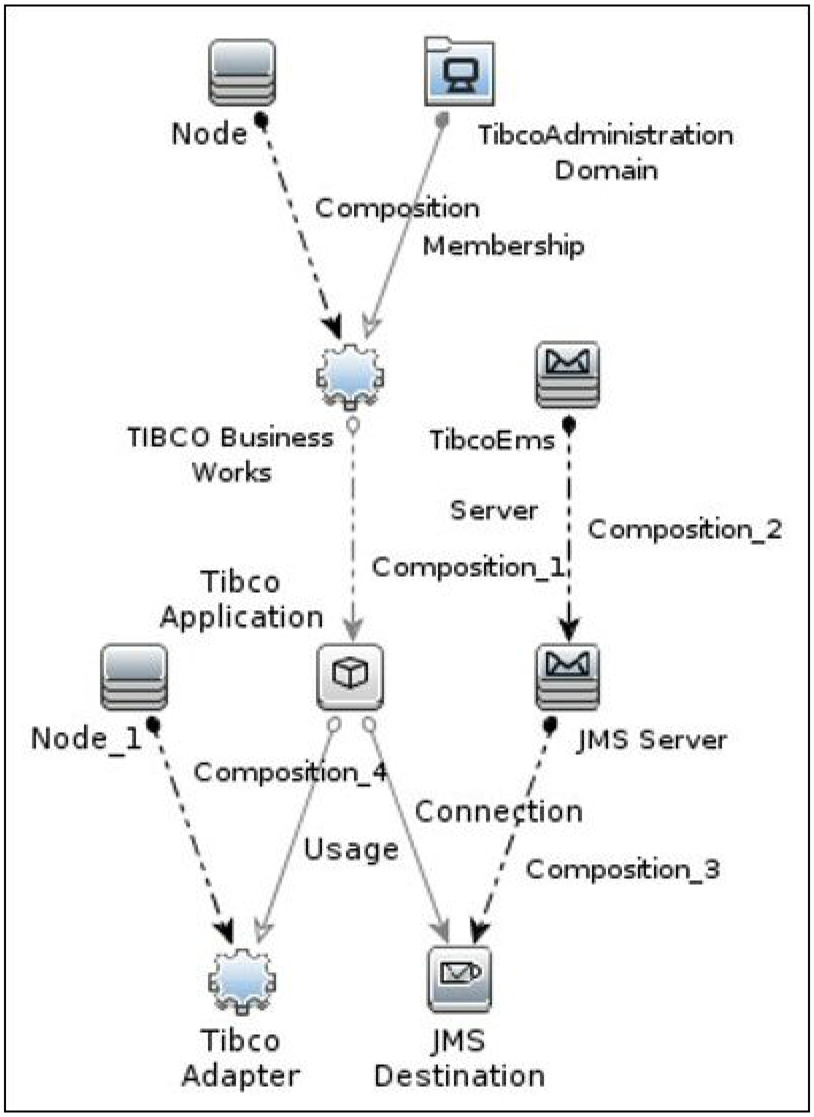
TIBCO discovery supports the following versions of software running in a UNIX environment:

- Version 6.0 of EMS
- Versions 5.7 and 5.8 of BusinessWorks.

Topology

The following image displays BusinessWorks topology.

Note: For a list of discovered CITs, see "TIBCO BusinessWorks by Shell Job" on page 873.



How to Discover TIBCO BusinessWorks and EMS

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   You must set up the Shell (SSH or Telnet) and TIBCO protocols.

   - Shell Protocols: SSH, Telnet.

     Prepare the following information: **user name**, **password** and **domain name**.

   - TIBCO Protocol

     Prepare the following information: **user name**, and **password**.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   a. Run the **Range IPs by ICMP** job in order to discover the target IPs.

   b. Run the **Host Connection by Shell** job in order to discover the target host and shell connectivity to it.

   c. Run the **Host Applications by Shell** job in order to discover applications of the target host, including TIBCO BusinessWorks software and agent processes.

   Note: You must enable the **discoverProcesses** attribute; this finds the **Process** CI on which the **TIBCO EMS by Shell** job triggers.

   d. Ensure you have **both** of the following:

      i. Read and write access to the temporary folder on the TIBCO BusinessWorks server. The default folder is **/tmp**.

      ii. Access to run the **TIBCO runtime assistant (TRA) AppManage** utility.

3. **Run the Discovery**

   a. Run the **TIBCO BusinessWorks by Shell** job in order to discover the topology of the target

BusinessWorks server.

b. Run the **TIBCO EMS by Shell** job in order to discover the topology of the target EMS server.

TIBCO BusinessWorks by Shell Job

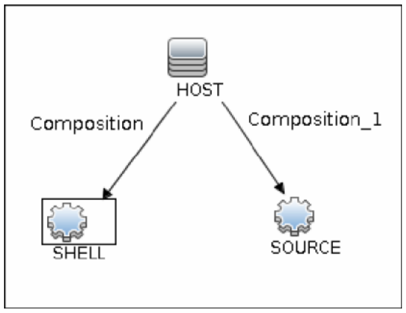
This section includes details about the job.

Input CIT

TibcoBusinessWorks

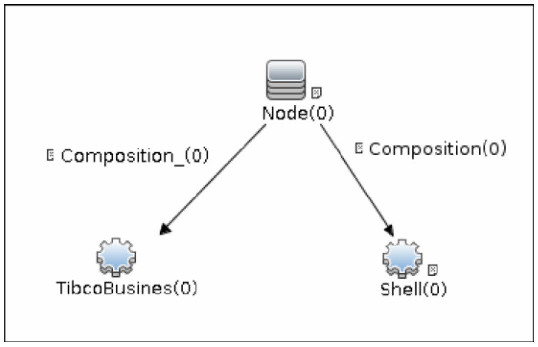
Input TQL Query

The following graphic shows an input TQL query for this job.



Trigger TQL Query

The following graphic shows a trigger TQL query for this job.



Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}

Name	Value
bwId	${SOURCE.root_id}
bwPath	${SOURCE.application_path}
credentialsId	${SHELL.credentials_id}
hostId	${HOST.root_id}
ip_address	${SHELL.application_ip}

Used Scripts

- db.py
- db_builder.py
- db_platform.py
- entity.py
- iteratortools.py
- jdb_url_parser.py
- jdbc.py
- jee.py
- jms.py
- jmx.py
- tibco.py
- tibco_businessworks_by_shell.py
- tibco_discoverer.py

Discovered CITs

- Composition

- Connection
- Containment
- IpAddress
- IpServiceEndpoint
- JMS Desination
- JMS Server
- Membership
- Node
- TibcoAdapter
- TibcoAdministrationDomain
- TibcoApplication
- TibcoBusinessWorks
- TibcoEmsServer
- Usage

Parameters

Parameter	Description
temp_directory	This is the temporary directory on the TIBCO BusinessWorks server where files created in the discovery process are stored.
discover_jms_topology	Whether or not to discover JMS topology. **Default:** false.

TIBCO EMS by Shell Job

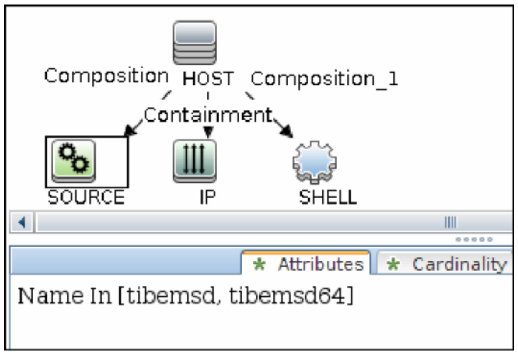
This section includes details about the job.

Input CIT

Process

Input TQL Query

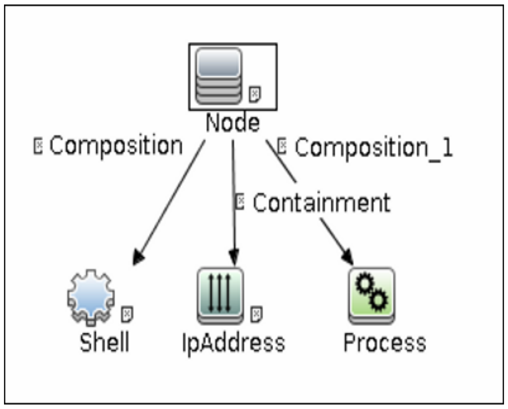
The following graphic shows an input TQL query for this job.



Name In [tibemsd, tibemsd64]

Trigger TQL Query

The following graphic shows a trigger TQL query for this job.



Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
hostId	${HOST.root_id}

Name	Value
ip_address	${SHELL.application_ip}
processCMdLine	${SOURCE.process_cmdline}
processPath	${SOURCE.process_path}
processRootId	${SOURCE.root_id}

Used Scripts

- db.py
- db_builder.py
- db_platform.py
- entity.py
- iteratortools.py
- jdb_url_parser.py
- jdbc.py
- jee.py
- jms.py
- jmx.py
- tibco.py
- tibco_discoverer.py
- tibco_ems_by_shell.py

Discovered CITs

- Composition
- Containment

- IpAddress
- IpServiceEndpoint
- JMS Destination
- JMS Server
- Node
- Process
- TibcoEmsServer
- Usage

Parameters

Parameter	Description
discover_queues	Whether or not to discover information about queues. **Default:** false.
discover_topics	Whether or not to discover information about topics. **Default:** false.

Chapter 64: WebSphere MQ Discovery

This chapter includes:

Overview

The WebSphere MQ package enables mapping the various components of WebSphere MQ infrastructure in an organization. The end goal is to model its interdependence with other applications or services within the organization and enable end to end impact analysis across the messaging silo.

Message Queuing is a middle-ware technology that enables disparate software services to communicate in a way that does not require any knowledge of the target service. Reliable communication can be achieved regardless of current availability of the target system or complexity of the infrastructure connecting the two systems.

A Message may contain simple character data, numeric data, complex binary data, a request for information, a command, or a mixture of all of these. The messaging infrastructure is responsible for reliable and transparent transportation of a message from the source to the target and is not required to understand or be aware of its content.

Supported Versions

- **Target Platform.** IBM WebSphere MQ
- **Target Platform Versions.** 5.x, 6.x, 7.1
- **Target Platform OS.** Microsoft Windows, Solaris, Linux, AIX

Topology

The WebSphere MQ package includes the following views that model details of the MQ infrastructure. Each view has a corresponding report with the same query configuration.

Note:

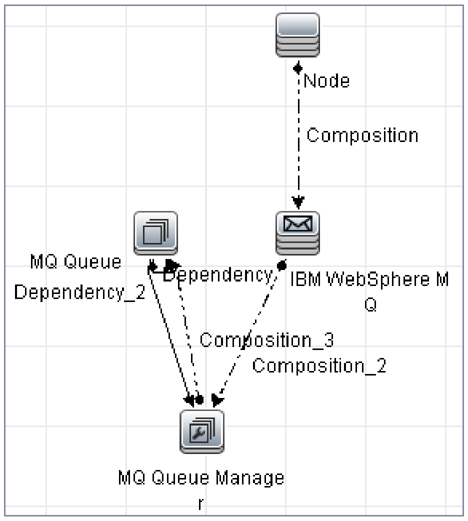
- These out-of-the-box views are provided as examples only. You may prefer to define your own views.
- For a list of discovered CITs, see "Discovered CITs" on page 889.

This section describes the following views:

- "MQ Queue Dependency" below
- "MQ Q Manager Resources on Non-Local Cluster" on the next page
- "MQ Namelist Membership" on page 883
- "MQ Cluster Membership" on page 883
- "MQ Channel Communication" on page 884
- "MQ Alias Queue Managers" on page 884
- "MQ Topology" on page 884

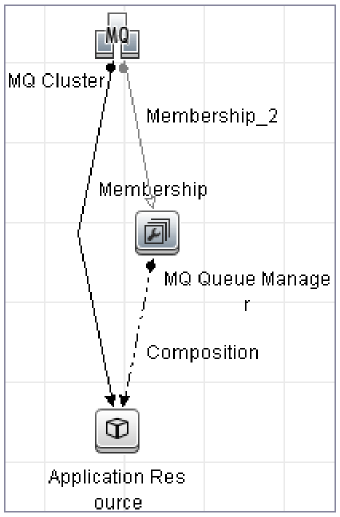
MQ Queue Dependency

This view displays queues that are dependent on other MQ objects and typically include Remote Queues, Alias Queues, and Remote Queue Managers:



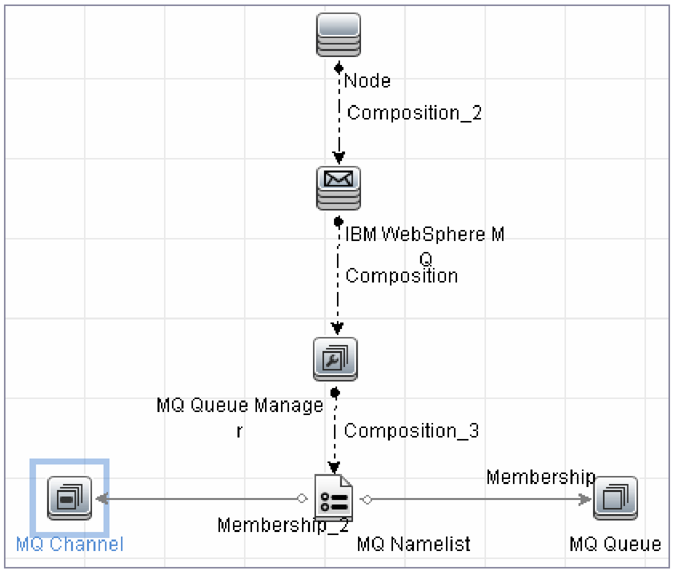
MQ Q Manager Resources on Non-Local Cluster

This view displays MQ objects managed by a Queue Manager and belonging to an MQ Cluster that the Queue Manager is not a member of. Any MQ objects in this view may be misconfigured and the purpose of this view is to identify such misconfigured objects.



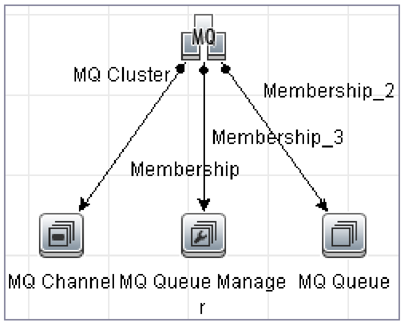
MQ Namelist Membership

This view displays namelists and their members:



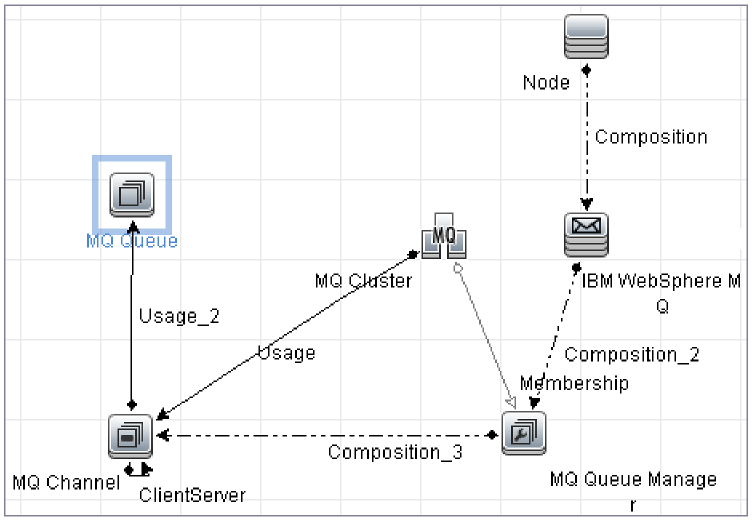
MQ Cluster Membership

This view displays clusters and their members:



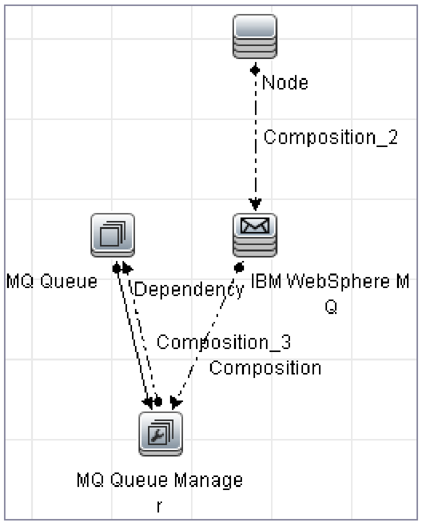
MQ Channel Communication

This view displays client-server communication between MQ Channels and queues used by the channels:



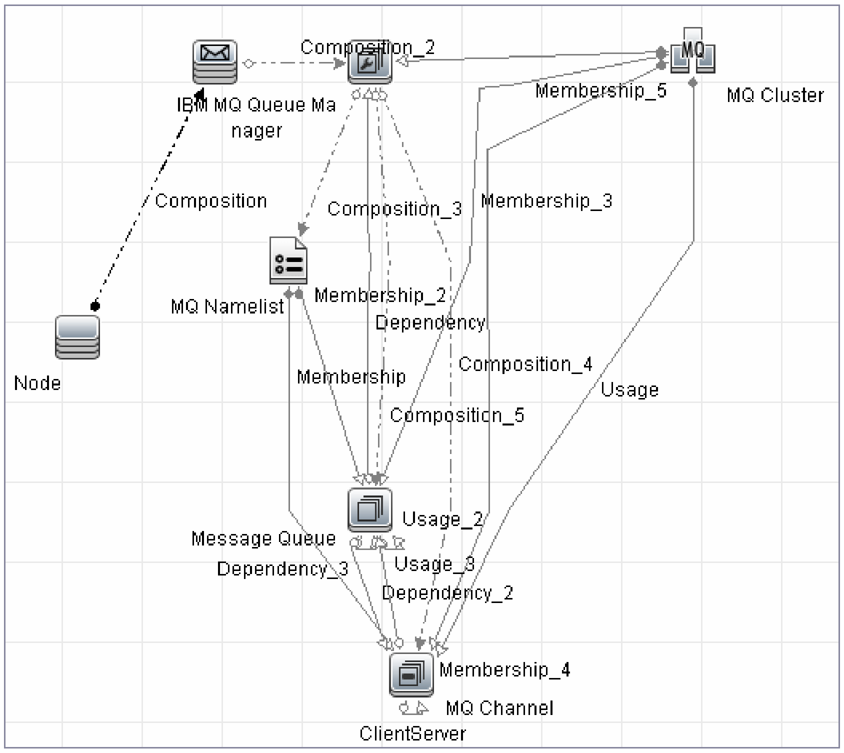
MQ Alias Queue Managers

This view displays Queues that are serving as remote Queue Managers:



MQ Topology

This view displays all MQ objects in the MQ infrastructure including relationships and interdependencies:


Composition_2
MQ
IBM MQ Queue Ma
nager
Membership_5
MQ Cluster
Composition
Composition_3
Membership_3
MQ Namelist
Membership_2
Dependency
Node
Composition_4
Membership
Usage
Composition_5
Usage_2
Message Queue
Dependency_3
Usage_3
Dependency_2
Membership_4
MQ Channel
ClientServer

How to Discover WebSphere MQ

The WebSphere MQ job discovers WebSphere MQ components and includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the SSH, Telnet, or NTCMD protocols.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   The Shell commands are (**sudo** is optional):

   - **dspmqver** or **mqver**
   - **dsmpq**
   - **runmqsc** or **runmqadm -r**

2. **Prerequisite - IP Addresses**

   Verify that all WebSphere MQ server IP addresses are within the scope of the Data Flow Probe. For details, see the section describing how to add Probe range in the *HP Universal CMDB Data Flow Management Guide*.

3. **Run the discovery**

   a. Configure parameters for the **MQ by Shell** job as necessary. For details, see "Data Flow Probe Setup" in the *HP Universal CMDB Data Flow Management Guide*.

   b. Run the following jobs to collect information required to trigger WebSphere MQ discovery:

      - **Range IPs by ICMP**. Discovers the WebSphere MQ server IP addresses.
      - **Host Connection by Shell**. Discovers operating system information on the WebSphere MQ servers.
      - **Host Applications by Shell**. Discovers instances of WebSphere MQ on the servers.
      - **MQ by Shell**. Discovers the WebSphere MQ infrastructure.

Discovery Mechanism

WebSphere MQ can be installed on several UNIX platforms and Microsoft Windows,and is managed using a command line interface standardized across platforms. The command line interface is accessible through programs, **runqsc** or **runmqadm,** that are included in a WebSphere MQ installation.

The **MQ by Shell** job uses the **Shell** CI associated with a server as its trigger. Because every server in the CMDB may have an associated **Shell** CI, the trigger query results contain the **Shell** CI only for servers on which WebSphere MQ software is installed.

The **MQ by Shell** job uses the WebSphere MQ command line interface to query for MQ objects and their details. Since the **runmqsc** command requires administrator or root privileges and the **runmqadm** command is not always available, the job attempts the **runmqadm -r** command first. If **runmqadm** fails, the job tries the **runmqsc** command.

After logging in to the MQ server using the **Shell** CI (created by the **Host Connections by Shell** job), DFM:

1. Identifies the version of WebSphere MQ installed on the server. This is done using the **dspmqver** command. (If **dspmqver** fails, the **mqver** command is attempted.)

2. Retrieves a list of WebSphere MQ Queue Managers using the **dspmq** command.

3. Retrieves details on each Queue Manager using the MQ CLI (command line interface) command:

   ```
   DISPLAY QMGR DESCR DEADQ DEFXMITQ REPOS CCSID
   ```

4. Retrieves a list of queues on each Queue Manager using the MQ CLI command:

   ```
   DISPLAY QUEUE(*) TYPE DESCR CLUSTER CLUSNL USAGE RNAME RQMNAME XMITQ TARGQ
   DEFTYPE
   ```

   Relationships between queues and other MQ objects such as other queues, Queue Managers, and so on, are built on the fly.

5. Retrieves (for each `TRANSMIT Queue` found) the remote server name and IP and port using the sender channel associated with the transmit queue. This is done using the MQ CLI command:

   ```
   DISPLAY CHANNEL(*) WHERE(xmitq EQ <transmitQueueName>) TYPE(SDR) CONNAME
   ```

6. Retrieves a list of channels on each Queue Manager using the MQ CLI command:

   ```
   DISPLAY CHANNEL(*) CHLTYPE TRPTYPE DESCR CLUSTER CLUSNL CONNAME XMITQ
   ```

Relationships between channels and other MQ objects such as other queues, channels, and so on, are built on the fly.

7. Retrieves a list of clusters that each Queue Manager is a member of, or knows about, using the MQ CLI command:

   ```
   DISPLAY CLUSQMGR(*) CONNAME QMTYPE
   ```

   Relationships between clusters and other clusters are built on the fly.

8. Retrieves the `namelists` that each Queue Manager is a member of, or knows about, using the MQ CLI command:

   ```
   DISPLAY NAMELIST(*) NAMES NAMCOUNT DESCR
   ```

Adapter

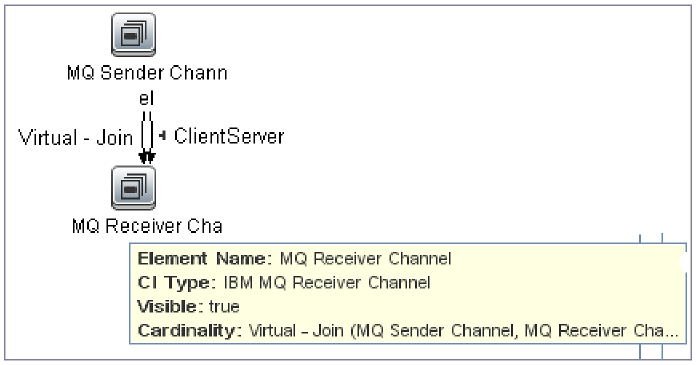
This discovery uses the **WebSphere MQ Topology by shell** adapter.

Adapter Parameters

Parameter	Description
discover_dynamic_queues	Enables discovery of dynamic queues (Queues created and destroyed on the fly by applications).
discover_remote_hosts	Enables resolution and discovery of remote servers and MQ objects referenced by the MQ server being discovered. If set to **false**, relationships between MQ objects on different servers are not discovered.
mq_cmd_timeout	Sets the command time-out for MQ CLI commands.
mqver_path	Path to **mqver** or **dspmqver** executable files. Separate multiple entries by a comma (**;**).
sudo_command	Must be set if the **use_sudo** parameter is set to **true**. Any entry here is prefixed to the MQ command line interface program. This parameter is typically used to set the MQ username. For example, if this parameter is set to **sudo -u mqm** the **runmqsc** command is invoked as **sudo -u mqm runmqsc**.
use_sudo	Set to **true** to enable sudo usage.

Enrichment Rule

The WebSphere MQ package includes an enrichment rule to link sender and receiver channels. The sender and receiver channels reside on different Queue Managers and have the same name.



Element Name: MQ Receiver Channel
CI Type: IBM MQ Receiver Channel
Visible: true
Cardinality: Virtual – Join (MQ Sender Channel, MQ Receiver Cha...

Discovered CITs

The WebSphere MQ discovery discovers the following CI Types.

Note: To view the topology, see "Topology" on page 880.

CI Type	Key Attributes	Description
IBM WebSphere MQ (webspheremq) Parent: Message Queuing Software	• Name: Always **IBM WebSphere MQ** • Container: Node	Represents an instance of WebSphere MQ software installed on a server.
IBM MQ Queue Manager (mqqueue) Parent: Message Queue Resource	• Name • Container: IBM WebSphere MQ CI	Represents an MQ Queue Manager. A WebSphere MQ instance may have one or more Queue Managers. The Queue Manager is responsible for functions not directly related to data movement such as storage, timing, triggering, and so on. Queue managers use a proprietary IBM technology known as a **bindings** connection to communicate with the MQ objects it manages and with remote clients via a network.

CI Type	Key Attributes	Description
IBM MQ Namelist (mqnamelist) Parent: Message Queue Resource	• Name • Container: IBM MQ Queue Manager	Represents an MQ Namelist. An MQ namelist contains a list of names and is typically used to contain a list of MQ Queue Manager Clusters. These namelists are then specified in the cluster namelist property and may be used by all Queue Managers in that cluster for look up.
IBM MQ Channel (mqchannel) Parent: Message Queue Resource	• Name • Container: IBM MQ Queue Manager	This abstract CI Type represents MQ Channels. MQ Channels are required by Queue Managers to communicate with other Queue Managers. Channels have uni-directional and bi-directional communication (such as a request-response system) and require a second channel to return data. A channel sends or receives data on a specific port on a TCP/IP network.
IBM MQ Cluster (mqcluster) Parent: Failover Cluster	Name	Represents an MQ Queue Manager Cluster An MQ Cluster provides a flexible approach to join multiple Queue Managers with minimal configuration. This enables multiple instances of the same service to be hosted through multiple Queue Managers, resulting in higher performance, capacity, and resiliency. Queue managers can dynamically join or leave clusters.
IBM MQ Queue (mqqueue) Parent: MQ Queue	• Name • Container: IBM MQ Queue Manager	A Queue is a container of messages in the MQ infrastructure and controls how messages are routed between Queue Managers in the MQ infrastructure. Queues may be set up in several configurations to control message ordering and delivery (F/LIFO, message priority, sequential delivery, guaranteed delivery, and so on) and are optimized to carry small amounts of information.
IBM MQ Alias Queue (mqlocalqueue) Parent: IBM MQ Queue	• Name • Container: IBM MQ Queue Manager	Represents MQ Alias Queues. An Alias Queue is an alias of another queue. It can be an alias of a local, remote, transmission, or another alias queue. The alias queue and the queue for which it is an alias are within the same Queue Manager. Messages and commands issued on the alias queue are forwarded to the queue for which it is an alias.

CI Type	Key Attributes	Description
IBM MQ Local Queue (mqlocalqueue) Parent: IBM MQ Queue	• Name • Container: IBM MQ Queue Manager	Represents MQ Local Queues. A Local Queue is a basic message queue and container of messages. An application can place a message in it for delivery or request, or retrieve a message from it.
IBM MQ Remote Queue (mqlocalqueue) Parent: IBM MQ Queue	• Name • Container: IBM MQ Queue Manager	Represents MQ Remote Queues. A Remote Queue is a remote or proxy instance of another queue. It can be a remote instance for a local, remote, transmission, or another alias queue. The remote queue and the queue for which it is a remote may be on different Queue Managers. A Remote Queue may also be a remote or proxy of a Queue Manager, and is represented as a remote Queue Manager.
IBM MQ Transmit Queue (mqlocalqueue) Parent: IBM MQ Queue	• Name • Container: IBM MQ Queue Manager	Represents MQ Transmission Queues. A Transmission Queue is a special purpose queue that transmits messages from one Queue Manager to another through MQ Channels. Remote queues use transmission queues to relay messages to the queue for which it is a remote.
IBM MQ Receiver Channel (mqreceiverchannel) Parent: IBM MQ Channel	• Name • Container: IBM MQ Queue Manager	A receiving channel receives messages from remote Queue Managers through a sending channel with the same name.
IBM MQ Sender Channel (mqsenderchannel) Parent: IBM MQ Channel	• Name • Container: IBM MQ Queue Manager	A sending channel is associated with a specific Transmission queue within the same parent Queue Manager and has a well-defined destination.

Relationships

WebSphere MQ discovery contains the following relationships:

Link	End1	End2	Cardinality	Description
Client Server	IBM MQ Send Channel	IBM MQ Receive Channel	1..*	Represents the direction of message flow between MQ Channels
Realization	IBM MQ Remote Queue	IBM MQ Queue	1..*	Indicates a strong dependency between an MQ Remote Queue and another Queue for which it is a remote. This is used in situations when the type of Queue is unknown.
Realization	IBM MQ Remote Queue	IBM MQ Local Queue	1..*	Indicates a strong dependency between an MQ Remote Queue and a Local Queue for which it is a remote.
Realization	IBM MQ Remote Queue	IBM MQ Alias Queue	1..*	Indicates a strong dependency between an MQ Remote Queue and an Alias Queue for which it is a remote.
Realization	IBM MQ Remote Queue	IBM MQ Remote Queue	1..*	Indicates a strong dependency between an MQ Remote Queue and a Remote Queue for which it is a remote.
Realization	IBM MQ Alias Queue	IBM MQ Queue	1..*	Indicates a strong dependency between an MQ Alias Queue and another Queue for which it is an alias. This is used in situations when the type of Queue is unknown.
Realization	IBM MQ Alias Queue	IBM MQ Local Queue	1..*	Indicates a strong dependency between an MQ Alias Queue and a Local Queue for which it is an alias.
Realization	IBM MQ Alias Queue	IBM MQ Remote Queue	1..*	Indicates a strong dependency between an MQ Alias Queue and a Remote Queue for which it is an alias.
Realization	IBM MQ Alias Queue	IBM MQ Alias Queue	1..*	Indicates a strong dependency between an MQ Alias Queue and an Alias Queue for which it is an alias.

Link	End1	End2	Cardinality	Description
Realization	IBM MQ Remote Queue	IBM MQ Queue Manager	1..*	Relates a queue of type remote queue (Remote Queue Manager) and the Queue Manager it is representing. This is a special purpose Remote Queue that is a remote for Queue Manager (instead of a remote queue). For Queue Managers QM1 and QM2, it is possible to set up a Remote Queue on QM1 named RQM2 which is a remote of QM2. Any MQ command issued to RQM2 is passed on to QM2 for execution.
Membership	IBM MQ Cluster	IBM MQ Queue Manager	1..*	Indicates that the MQ Queue Manager is a member of the MQ Queue Manager Cluster. If an MQ Queue Manager is a `full repository` for a cluster, the name of this relationship is set to **Repository**.
Membership	IBM MQ Cluster	IBM MQ Channel	1..*	Indicates that the MQ Channel is a member of the MQ Queue Manager Cluster. When a queue or channel is defined in any Queue Manager, it is possible (but not necessary) to specify of which MQ cluster this queue is a member. This is useful when very specific configurations are required, for example, when a queue is a member of a cluster but the Queue Manager is not a member of that cluster. This link is used to identify these special configurations.
Membership	IBM MQ Cluster	IBM MQ Queue	1..*	Indicates that the MQ Queue is a member of the MQ Queue Manager Cluster. This link is added for the same reason as in the previous row.
Membership	IBM MQ Namelist	IBM MQ Channel	1..*	Indicates that the MQ Channel contains the name of the MQ Namelist in its `CLUSNL` parameter.
Membership	IBM MQ Namelist	IBM MQ Queue	1..*	Indicates that the MQ Queue contains the name of the MQ Namelist in its `CLUSNL` parameter.

Link	End1	End2	Cardinality	Description
Usage	IBM MQ Cluster	IBM MQ Channel	1..*	Indicates the MQ Channel (of types Cluster Sender Channel or Cluster Receiver Channel) used by the MQ Queue Manager Cluster for communication with another cluster. This relationship is specific to MQ Channels of type `Cluster Sender Channel` and `Cluster Receiver Channel`. These channels are dedicated to inter-cluster communication and are not used by queues or other MQ objects.
Usage	IBM MQ Remote Queue	IBM MQ Transmit Queue	1..*	Indicates a remote queue using a transmission queue for communication.
Usage	IBM MQ Transmit Queue	IBM MQ Sender Channel	1..*	Indicates a sender Transmission Queue using a Sender channel for communication.

Troubleshooting and Limitations

- If there are DNS resolution errors in the log files, and discovery takes abnormally long to complete, try setting the **discovery_remote_hosts** parameter to **false**. For details, see "Adapter Parameters" on page 888.
- If the discovery results appear incomplete, try increasing the value of the **mq_cmd_timeout** parameter. For details, see "Adapter Parameters" on page 888.

Part 12: Middleware > Web Servers

Chapter 65: Microsoft Internet Information Services (IIS) Discovery

This chapter includes:

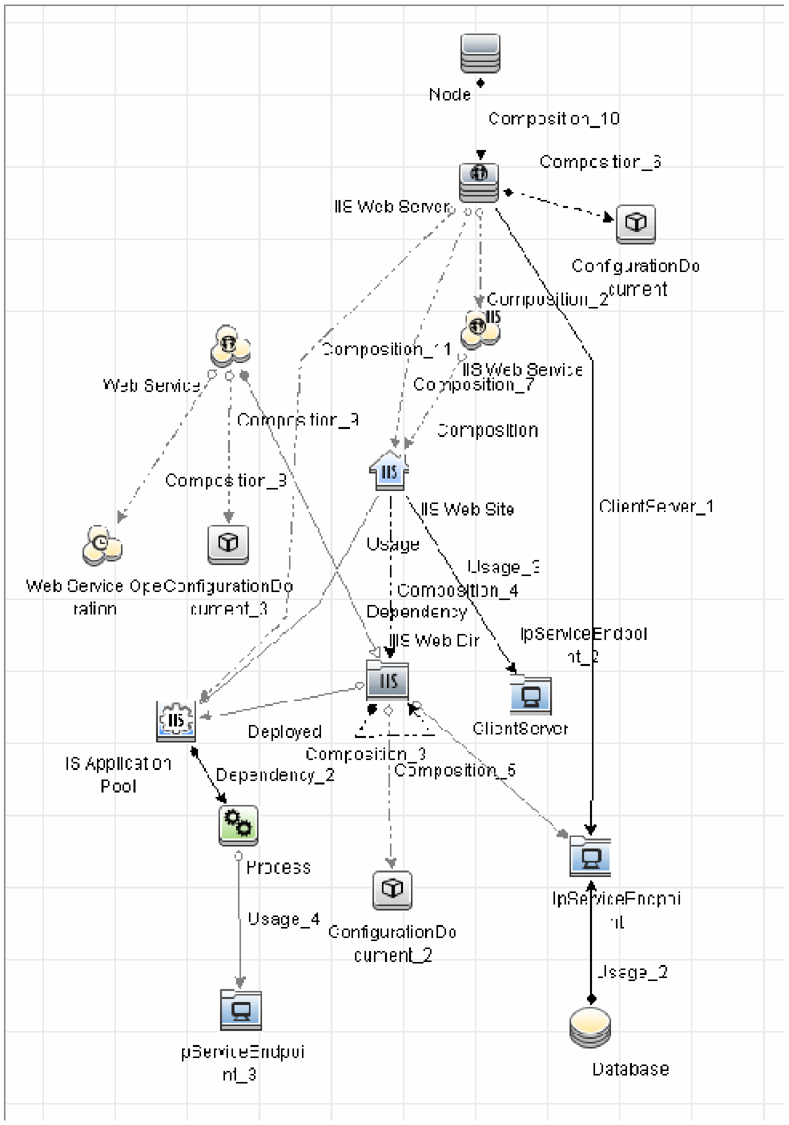
Supported Versions

This discovery supports Microsoft Internet Information Services (IIS) versions: 5, 6, 7.

Note: Discovery of IIS 7 is supported through the IIS 6 Management Compatibility tool. This tool must be installed to perform discovery of IIS 7.

Microsoft Internet Information Services (IIS) Discovery Topology

Note: For a list of discovered CITs, see "IIS Applications by NTCMD or UDA Job" on page 900.



How to Discover Microsoft Internet Information Services (IIS) Topology

This task describes how to discover Microsoft Internet Information Services (IIS) and includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the **NTCMD** protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Other**

   - To retrieve all relevant information, DFM should be able to execute Visual Basic scripts and have write permission to the **%SystemRoot%/
     system32/drivers/etc** folder.

   - Verify that the target machine running IIS lies in the Data Flow Probe range.

3. **Run the discovery**

   In the Universal Discovery window, activate the jobs in the following order:

   a. Run the **Host Connection by Shell** job to create Shell CITs.

   b. Run the **Host Applications by Shell** job to discover IIS Web Server CIs and IIS Application Pool CIs with corresponding **Depend** links to the managing process.

   c. Run the **IIS Applications by NTCMD or UDA** job to discover the detailed topology of IIS.

      After the connection is made, DFM copies the **adsutil.vbs** script on the remote machine. DFM retrieves IIS topology information from the output of this tool.

      Microsoft IIS version 7.0 enables you to create an IIS application from a Web directory, as well as from a virtual directory (as in prior versions). Therefore, when DFM discovers such an application, DFM creates an IIS Web Directory CI.

      To view required permissions: **Universal Discovery > Discovery Modules/Jobs > Middleware > Web Servers > IIS > IIS Applications by NTCMD or UDA** job. **Details** tab > **Discovery Job**

Details pane. Click the **View Permissions** button. For details, see "IIS Applications by NTCMD or UDA Job" below.

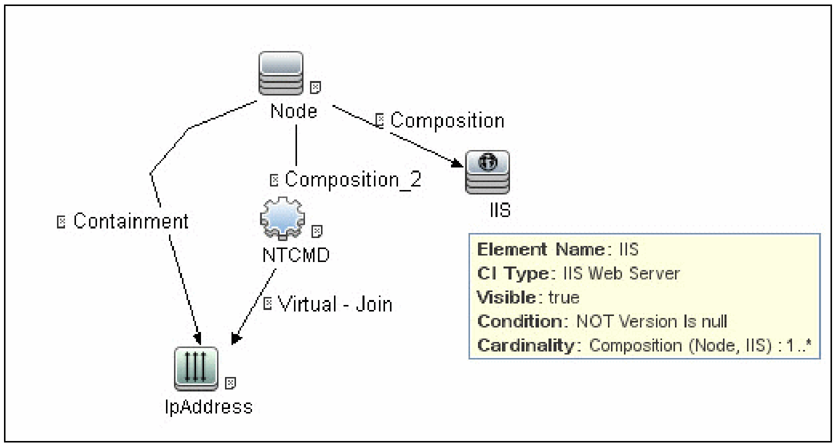
Note: The IIS Web Dir CI is created only if there is an `IIS Virtual Dir` CI or a `web.config` file underneath in the topology, otherwise it is not reported.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide.*

IIS Applications by NTCMD or UDA Job

This section includes:

Trigger Query



Element Name: IIS
CI Type: IIS Web Server
Visible: true
Condition: NOT Version Is null
Cardinality: Composition (Node, IIS) : 1..*

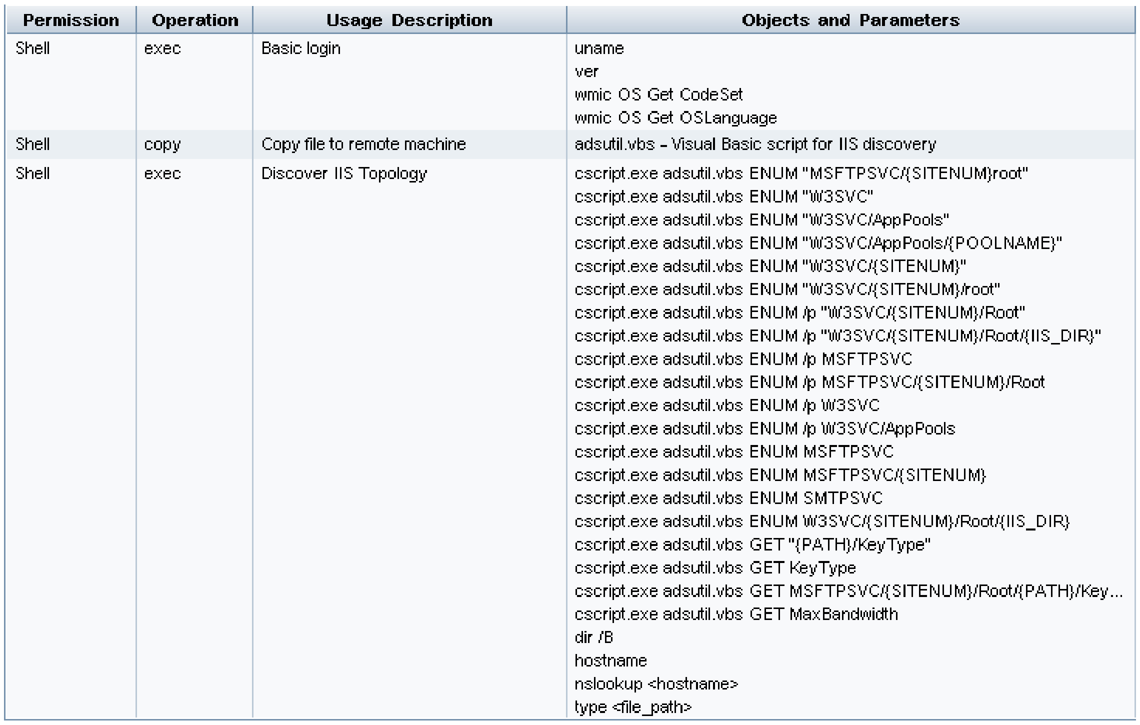
Adapter

This job uses the NTCMD_APP_Dis_IIS adapter.

- Triggered CI Data

Name	Value
credentialsId	${NTCMD.credentials_id}
iis_name	${SOURCE.discovered_product_name}
iis_version	${SOURCE.version}
ip_address	${NTCMD.application_ip}

- Permissions



Permission	Operation	Usage Description	Objects and Parameters
Shell	exec	Basic login	uname ver wmic OS Get CodeSet wmic OS Get OSLanguage
Shell	copy	Copy file to remote machine	adsutil.vbs – Visual Basic script for IIS discovery
Shell	exec	Discover IIS Topology	cscript.exe adsutil.vbs ENUM "MSFTPSVC/{SITENUM}root" cscript.exe adsutil.vbs ENUM "W3SVC" cscript.exe adsutil.vbs ENUM "W3SVC/AppPools" cscript.exe adsutil.vbs ENUM "W3SVC/AppPools/{POOLNAME}" cscript.exe adsutil.vbs ENUM "W3SVC/{SITENUM}" cscript.exe adsutil.vbs ENUM "W3SVC/{SITENUM}/root" cscript.exe adsutil.vbs ENUM /p "W3SVC/{SITENUM}/Root" cscript.exe adsutil.vbs ENUM /p "W3SVC/{SITENUM}/Root/{IIS_DIR}" cscript.exe adsutil.vbs ENUM /p MSFTPSVC cscript.exe adsutil.vbs ENUM /p MSFTPSVC/{SITENUM}/Root cscript.exe adsutil.vbs ENUM /p W3SVC cscript.exe adsutil.vbs ENUM /p W3SVC/AppPools cscript.exe adsutil.vbs ENUM MSFTPSVC cscript.exe adsutil.vbs ENUM MSFTPSVC/{SITENUM} cscript.exe adsutil.vbs ENUM SMTPSVC cscript.exe adsutil.vbs ENUM W3SVC/{SITENUM}/Root/{IIS_DIR} cscript.exe adsutil.vbs GET "{PATH}/KeyType" cscript.exe adsutil.vbs GET KeyType cscript.exe adsutil.vbs GET MSFTPSVC/{SITENUM}/Root/{PATH}/Key... cscript.exe adsutil.vbs GET MaxBandwidth dir /B hostname nslookup <hostname> type <file_path>

- Adapter Parameters

Parameter	Description
acceptedStatusCodes	Contains status code which should be treated as **OK** during the verification of URL.
adsutil_path	Enter the path and name to the **adsutil.vbs** script. The **adsutil.vbs** script is a free script provided by Microsoft for IIS management tasks.

Parameter	Description
checkConnectionToUrl	When set to **true**, any reported URL is verified on the availability by HTTP (s) head method from the probe machine. In case of an unsuccessful connection, the URL is skipped.
do_web_service	**True.** The IIS Web Service CI is reported. **Note: report_legacy_topology** must also be set to **true** for DFM to report this CI.
report_legacy_ topologyT	For backwards compatibility, DFM continues, by default, to report the legacy IIS topology.
web_service_file_ extensions	List of file extensions which will detect as web services. **Note:** Wildcards are not supported.

Discovered CITs

- ClientServer
- Composition
- ConfigurationDocument
- Containment
- Depedency
- Deployed
- IIS FTP Server
- IIS Resource
- IIS SMTP Server
- IIS Web Server
- IpAddress

- IpServiceEndpoint
- Node
- UriEndpoint
- Usage
- Web Server Virtual Host

Bugzilla, Wordpress, and MediaWiki Discovery

For details, see "How to Discover Bugzilla, Wordpress, and MediaWiki" on page 766.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Microsoft Internet Information Services (IIS) discovery.

- An IIS Web server CI is created even if no Web service is running on the machine but the IIS FTP and IIS SMTP services are present.
- If the discovered web.config file's ConnectionStrings property contains a password, when the configuration file CI is created the password is replaced with asterisk characters.

Part 13: Middleware > Web Services

Chapter 66: UDDI Registry Discovery

This chapter includes:

Overview

The UDDI discovery process enables you to discover Web services from a UDDI registry.

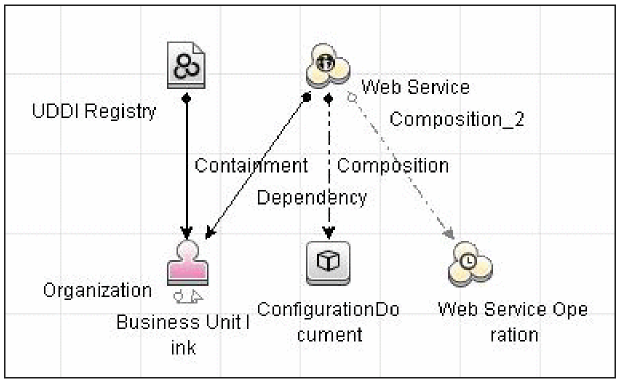
DFM queries the UDDI registry for its Web services, including non-SOAP services, or for a specific publisher service (if defined in the UDDI Registry protocol). The Web services found in the UDDI registry are represented by a **WebService Resource** CI in the CMDB and the registry is created as a **UDDI Registry** CI.

Supported Versions

This discovery supports UDDI versions 2 and 3.

Topology

The following depicts the topology of the **SOA_UDDI_View**:



How to Discover UDDI Processes

This task includes the following steps:

1. **Prerequisites - Install ruddi jar file**

   a. Download the ruddi-1.0-bin.jar file from the following location:

      http://sourceforge.net/projects/s-feng/files/S-FENG/lib/ruddi1.0/ruddi-1.0-bin.jar/download?use_mirror=iweb

   b. Rename the file **ruddi.jar** and copy it to the following directory on the probe machine:

      <hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\uddi\

2. **Prerequisites - Set up protocol credentials**

   Set up the UDDI protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

3. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Activate the following jobs:

   - **Web Services by URL**
   - **Web Service Connections by UDDI Registry**
   - **Web Services by UDDI Registry**

4. **Provide service publisher details – Optional**

   Update the UDDI Registry adapter's **organization** parameter with the name of the service publisher and a description of the organization.

   For more details about editing adapter parameters, see "Adapter Definition Tab" in the *HP Universal CMDB Data Flow Management Guide*.

Part 14: Middleware > Proxy Servers

Chapter 67: IBM Security Access Manager Discovery

This chapter includes:

Overview

This package discovers IBM Security Access Manager (previously called IBM Tivoli Access Manager), which is an integrated access appliance that provides web access security protection. This package includes adapters that discover this application using the following protocols:

- HTTP/Web
- Shell

Supported Versions

The following versions are supported:

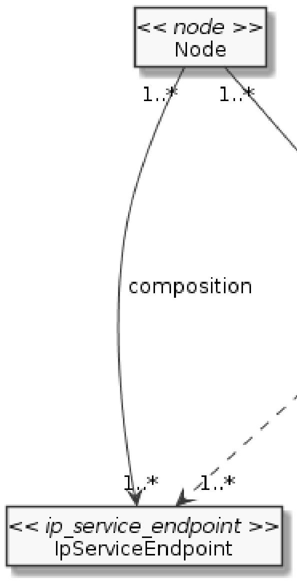
Job	Versions Supported
IBM Security Access Manager for Web by HTTP	8.x
IBM Security Access Manager for Web by Shell	6-8.x

WebSeal Connection By Web Services Job

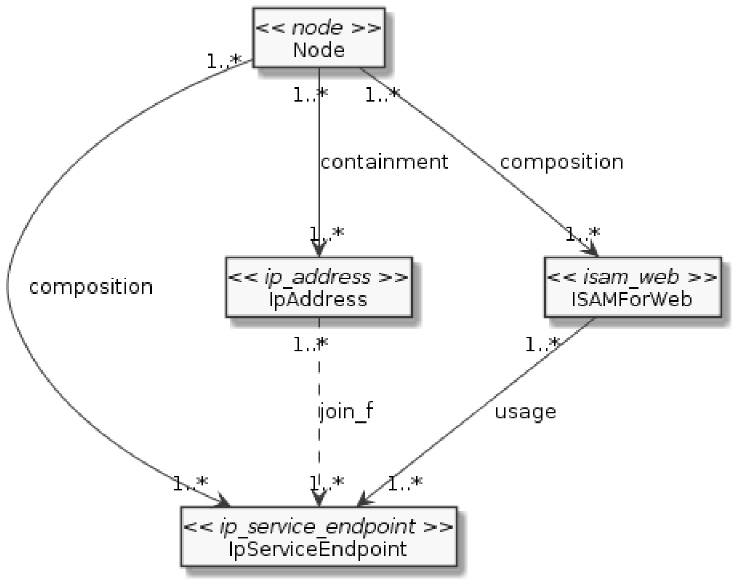
This section includes details about the job.

Trigger Query

- **https_ports.xml**



- **isam_with_ipse.xml**



Adapter

This job uses the **webseal_connection_by_webservices** adapter.

Used Scripts

- webseal_connection_by_webservices.py

Discovered CITs

- Composition
- Containment
- ISAMForWeb
- IpAddress
- IpServiceEndpoint
- Node
- Usage

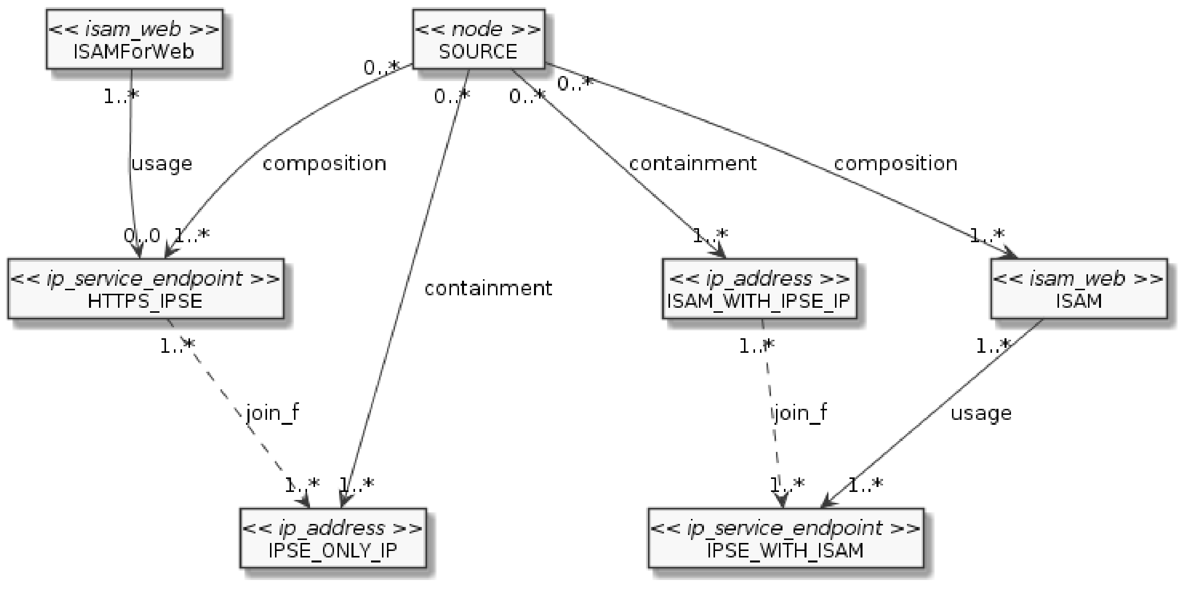
Parameters

Name	Type	Description
autoAcceptCerts	string	Temporary accept received certificates as trusted
firmware_settings_api_query	string	Http query to firmware settings web services api
management_authentication_api_query	string	Http query to management authentication web services api
pdadmin_api_query	string	Http query to pdadmin web services api
reverseproxy_api_query	string	Http query to reverseproxy web services api

Input CIT

- Node

Input TQL



Triggered CI Data

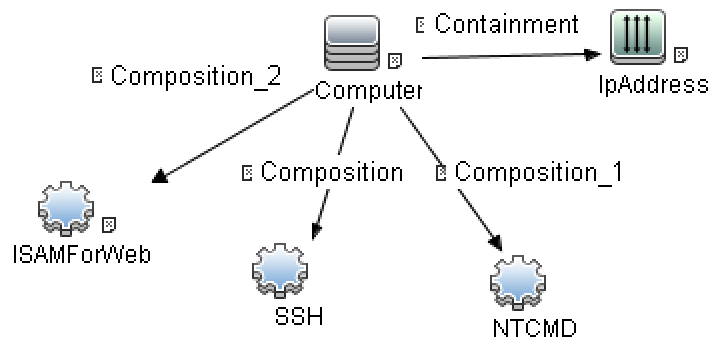
Name	Value	Description
https_ipse_ids	${HTTPS_IPSE.root_id:NA}	List of root ids of IpServiceEndpoint in case when trigger is https IpServiceEndpoint without ISAM instance

Name	Value	Description
ipse_only_ ips	${IPSE_ONLY_ IP.name:NA}	List of ip addresses in case when trigger is https IpServiceEndpoint without ISAM instance
isam_ credential_ ids	${ISAM.credentials_ id:NA}	List of credential id in case when trigger is ISAM instance
isam_ids	${ISAM.root_id:NA}	List of ISAM root ids in case when trigger is ISAM instance
isam_with_ ipse_ips	${ISAM_WITH_ IPSE_IP.name:NA}	List of ip addresses in case when triger is ISAM instance

WebSeal Connection By Shell Job

This section includes details about the job.

Trigger Query



Adapter

This job uses the **webseal_connection_by_shell** adapter.

Used Scripts

- webseal_topology.py

- pdadmin_shell_webseal_discoverer.py
- webseal_connection_by_shell.py

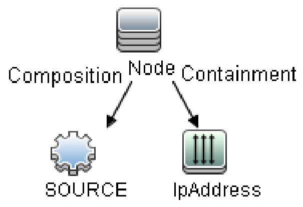
Discovered CITs

- Composition
- Containment
- ISAMForWeb
- IpAddress
- Node

Input CIT

- Shell

Input TQL



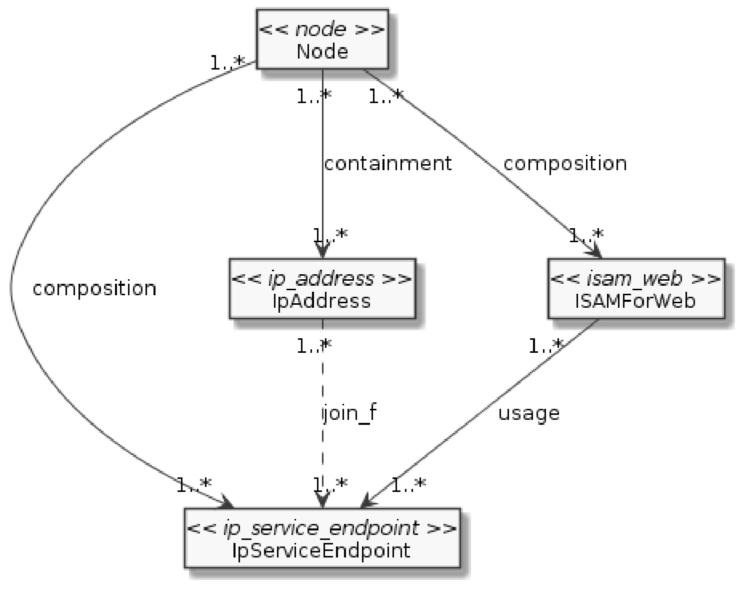
Triggered CI Data

Name	Value	Description
Protocol	${SOURCE.root_class}	shell type (ssh/ntcmd)
credentialId	${SOURCE.credentials_id}	valid shell credentials id
hostId	${SOURCE.root_container}	Node UCMDB Id which Webseal belongs to
ip_address	${SOURCE.application_ip}	destination ip address

WebSeal Topology by Web Services Job

This section includes details about the job.

Trigger Query



Adapter

This job uses the **webseal_topology_by_webservices** adapter.

Used Scripts

- webseal_topology_by_webservices.py

Discovered CITs

- Composition
- Containment
- DirectoryServer
- ISAMForWeb

- ISAMJunction
- ISAMPolicyServer
- IpAddress
- IpServiceEndpoint
- Node
- RunningSoftware
- Usage

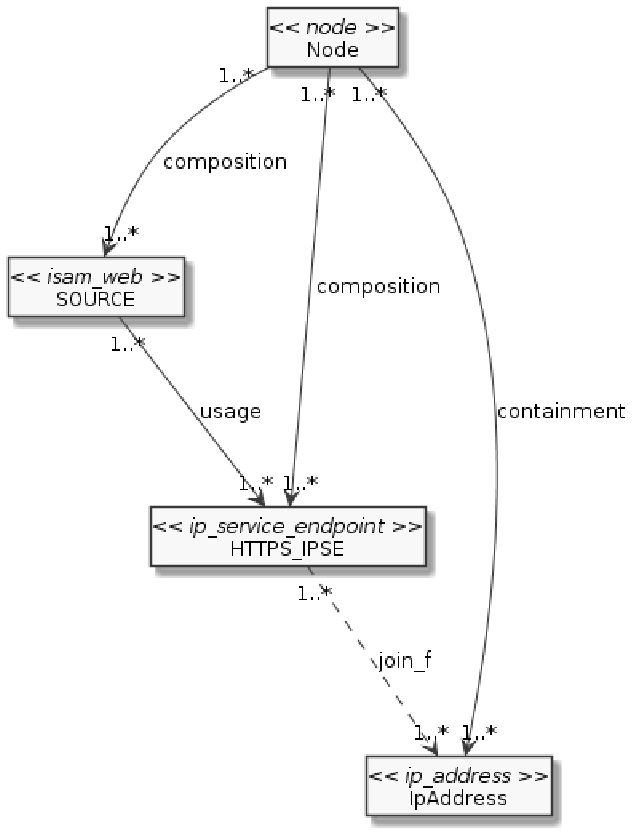
Parameters

Name	Type	Description
autoAcceptCerts	string	Temporary accept received certificates as trusted
firmware_settings_api_query	string	Http query to firmware settings web services api
management_authentication_api_query	string	Http query to management authentication web services api
pdadmin_api_query	string	Http query to pdadmin web services api
reverseproxy_api_query	string	Http query to reverseproxy web services api

Input CIT

- ISAMForWeb

Input TQL



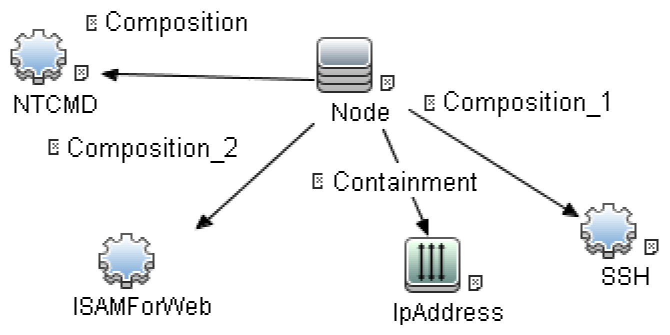
Triggered CI Data

Name	Value	Description
ip_address	${IpAddress.name}	List of ip addresses of trigger ISAM instance
container_cmdbid	${Node.root_id}	Root id of a container
cmdbid	${SOURCE.root_id}	Root id of ISAM instance
credential_id	${SOURCE.credentials_id}	Reference to credentials dictionary
name	${SOURCE.name}	Name of ISAM instance

WebSeal Topology by Shell Job

This section includes details about the job.

Trigger Query



Adapter

This job uses the **webseal_topology_by_shell** adapter.

Used Scripts

- webseal_topology_by_shell.py
- pdadmin_shell_webseal_discoverer.py
- webseal_topology.py

Discovered CITs

- Composition
- Containment
- DirectoryServer
- ISAMForWeb
- ISAMJunction
- ISAMPolicyServer
- IpAddress
- IpServiceEndpoint

- Node
- RunningSoftware
- Usage

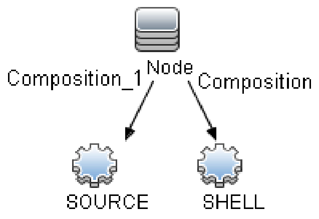
Parameters

Name	Type	Description
autoAcceptCerts	string	Temporary accept received certificates as trusted
firmware_settings_api_query	string	Http query to firmware settings web services api
management_authentication_api_query	string	Http query to management authentication web services api
pdadmin_api_query	string	Http query to pdadmin web services api
reverseproxy_api_query	string	Http query to reverseproxy web services api

Input CIT

- ISAMForWeb

Input TQL



Triggered CI Data

Name	Value	Description
Protocol	${SHELL.root_class}	shell type (ssh/ntcmd)
credentialId	${SHELL.credentials_id}	valid shell credentials id
hostId	${SHELL.root_container}	Node UCMDB Id which Webseal belongs to
ip_address	${SHELL.application_ip}	destination ip address

Name	Value	Description
webseal_credentials_id	${SOURCE.isam_credentials_id}	Reference to credentials dictionary

How to Discover IBM Security Access Manager Using HTTP

This task describes how to discover IBM Security Access Manager. It includes the following steps:

- "Prerequisite - Set up protocol credentials" on page 644
- "Prerequisites - Other" on page 644
- "Run the discovery" on page 645

Prerequisite - Set up protocol credentials

Set up the following protocols:

Job	Protocol
IBM Security Access Manager by HTTP	HTTP

Prerequisites- Other

Policy server account availability

HTTP-based discovery is based on Webservices API provided by the WebSeal platform. It uses pdadmin webservices API, and this API requires Policy Server credentials to be configured. Discovery uses the same username and password that were used for the basic authentication. This means that there should be available LDAP user with the same username/password which are configured to access WebSeal management console.

Run the discovery

Activate the following jobs:

- Range IP by ICMP
- Databases TCP Ports

- Webseal Connection By Web Services
- Webseal Topology By Web Services

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover IBM Security Access Manager Using Shell

This task describes how to discover IBM Security Access Manager. It includes the following steps:

- "Prerequisite - Set up protocol credentials" on page 644
- "Prerequisites - Other" on page 644
- "Run the discovery" on page 645

Prerequisite - Set up protocol credentials

Set up the following protocols:

Job	Protocol
IBM Security Access Manager by Shell	SSH or NTCMD Optional: LDAP for deep topology discovery.

Prerequisites- Other

Policy server account availability

Shell-based discovery connects to remote nodes and then uses the pdadmin utility. This utility requires an LDAP account that has the same credentials as the WebSeal account.

Run the discovery

Activate the following jobs:

- Range IP by ICMP
- Host Connection by Shell

- Host Applications by Shell
- Webseal Connection By Shell
- Webseal Topology By Shell

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Part 15: Network Infrastructure

Chapter 68: Network - Basic Discovery

This chapter includes:

Overview

You activate the jobs in the network modules to establish a Shell connection to host machines. Discovery tries to connect to the remote machine through the SSH, Telnet, and NTCMD protocols, until the first valid connection is found.

The module includes the following jobs:

- **Host Connection by Shell**. Establishes the connection to remote machines through the SSH, Telnet, NTCMD, and Universal Discovery protocols. This job discovers host type, OS information, and network connectivity information. For details, see "How to Discover Host Connection by Shell" on the next page.

- **Host Connection by SNMP**. Discovers SNMP agents by trying to connect, using the SNMP protocol, to a data center machine (whose IP addresses have previously been discovered and populated in IpAddress CIs in UCMDB). It then updates the correct host class (Windows, UNIX, router, and so on) according to the relevant OID. For details, see "How to Discover Host Connection by SNMP" on page 929.

- **Host Connection by WMI**. Establishes the connection to remote machines through the WMI protocol and discovers host type, OS information, and network connectivity information. For details, see "How to Discover Host Connection by WMI" on page 931.

- **Client Connection by SNMP**. Discovers SNMP agents by running a ping sweep of all client ranges configured in the Data Flow Probe (or Management Zone). If successful, it connects to the IP address using the SNMP protocol, and updates the correct host class (Windows, UNIX, router, and so on) according to the relevant OID. For details, see "How to Discover Client Connection by SNMP" on page 931.

For details on using a wizard to discover the network, see "Infrastructure Discovery Wizard" in the *HP UCMDB Universal Discovery Content Guide - Discovery Activities* document.

For information about each job's discovery mechanism, see:

- **Host Connection by Shell.** "Discovery Mechanism" on page 932
- **Host Connection by SNMP**. "Discovery Mechanism" on page 945
- **Host Connection by WMI**. "Discovery Mechanism" on page 950
- **Client Connection by SNMP**. "Discovery Mechanism" on page 955

How to Discover Host Connection by Shell

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the following protocols:

   - NTCMD protocol
   - SSH protocol
   - Telnet protocol
   - UDA protocol

   Note: To discover Windows machines running an SSH server, set the **Shell Command Separator** attribute of the protocol to **AutoDetect**.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Host Connection by Shell job**

   When running the **Host Connection by Shell** job to discover Windows machines on which an SSH server running the F-Secure application is installed, you must make the following modifications to F-Secure:

   - Stop the F-Secure service completely.
   - Verify that there are no F-Secure leftover processes still running (**fssh*** processes).
   - Alter the following lines in the **sshd2_config** file. This is an F-Secure configuration file that resides in the F-Secure installation directory.
     - The **DoubleBackspace** setting should contain a **no** value, that is, `DoubleBackspace no`.
     - The **EmulationType** setting should contain a **raw** value, that is, `EmulationType raw`.
     - The **EmulationTypeForCommands** setting should contain a **raw** value, that is, `EmulationTypeForCommands raw`.

- Save the altered **sshd2_config** file.
- Restart the F-Secure service.

> **Note:** The Data Flow Probe enables an SSH-based connection to remote Windows machines only if the remote SSH server providers are **Open-SSH** or **F-Secure**.
>
> For **Open-SSH** (that provides SSH servers for the Windows, UNIX, and Linux operating systems), DFM supports connections to Open-SSH only if the Open-SSH version is later than, or equal to, 3.7.1 (for any operating system).

3. **Run the discovery**

   Run the **Host Connection by Shell** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Note: The Data Flow Probe enables an SSH-based connection to remote Windows machines only if the remote SSH server providers are **Open-SSH** or **F-Secure**.

For **Open-SSH** (that provides SSH servers for the Windows, UNIX, and Linux operating systems), DFM supports connections to Open-SSH only if the Open-SSH version is later than, or equal to, 3.7.1 (for any operating system).

How to Discover Host Connection by SNMP

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the SNMP protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document..

2. **Run the discovery**

   Run the **Host Connection by SNMP** job.

For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Host Connection by WMI

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the WMI protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document..

2. **Run the discovery**

   Run the **Host Connection by WMI** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Client Connection by SNMP

This task includes the following steps:

1. **Prerequisites - Set up protocol credentials**

   This discovery uses the SNMP protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   Run the **Client Connection by SNMP** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Host Connection by Shell Job

This subject includes the following sections:

Discovery Mechanism

This part of the discovery depends on whether you are discovering components installed on Windows machines, UNIX-based machines, or Nexus machines. For details on the DFM processes, see:

- "Windows Processes" on the next page
- "UNIX-Based Processes" on page 935
- "Nexus Processes" on page 940

Note:

- DFM runs through the credentials defined for the protocol and tries to connect successfully through one of them. Whenever there is a successful connection, the discovery remembers the last successful credential and caches it for reuse the next time the discovery is run.
- If the credentials (last used for this destination) do not exist, DFM iterates through the list of all configured shell credentials.

  DFM uses the following flow for this iteration: SSH, then Telnet, then NTCMD protocol credentials, to try to connect to the discovered destination.

  DFM skips credential entry if the IP Address of the discovered destination is outside the IP range scope of the credential.

DFM immediately stops using a protocol for the discovered destination if:

- There is no agent on the remote machine
- Connection is refused
- Connection times out
- There is an IO exception on opening a socket

DFM stops iteration through the list of configured credentials if:

- It successfully establishes connection
- It fails to connect to the discovered destination after trying all configured credentials

Windows Processes

This section describes the part of the workflow that DFM performs for discovering components residing on Windows machines.

1. DFM discovers host attributes (OS name, version, build number, service pack, installation type). DFM starts by using the first instruction in the following list to discover the host attributes. If that fails, DFM continues to the next:

   a. WMIC "OS" object;

      Full command:

      ```
      'wmic os get caption, otherTypeDescription, version, buildnumber,
      csdversion /format:list < %SystemRoot%\win.ini'
      ```

   b. Windows registry;

      Full query:

      ```
      HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion VER
      command; %SYSTEMROOT%\system32\prodspec.ini processing
      ```

2. Define BIOS UUID (**wmic**)

   Full command:

```
'wmic path win32_ComputerSystemProduct get uuid /format:list <
%SystemRoot%\win.ini'
```

3. Define the default gateway (**netstat**).

   Full command:

   ```
   'netstat -r -n'
   ```

4. Define the DNS server IPs (**ipconfig**).

5. Define the boot date.

   Full command:

   ```
   'wmic OS Get LastBootUpTime /format:list < %SystemRoot%\win.ini'
   ```

6. Define the network interfaces. The **wmic** command is used first because it retrieves more information about the interface. If that fails, the output of the **ipconfig** command is used.

   a. Querying NICCONFIG object we get information about MAC address, IP addresses, interface description, subnet IPs, dynamic or static flag.

      Full command:

      ```
      'wmic nicconfig where "MACAddress <> NULL" get
      IPAddress,MACAddress,IPSubnet,Description,DhcpEnabled /format:list <
      %SystemRoot%\\win.ini'
      ```

   b. IP filtering. Malformed and local IPs are ignored.

7. DFM checks whether the destination IP is local. If it is, DFM reports the host and IP only. If it is not local:

   a. DFM reports network interfaces apart from:

      - Interfaces that do not have a MAC address
      - Interfaces that belong to one of the following types: loopback, wireless, virtual, WAN miniport, RAS ASYNC, Bluetooth, FireWire, VPN, IPv6 tunneling.
      - The VMware interface, if **ignoreVmwareInterfaces** is set to **true** in the **globalSettings.xml** configuration file.

   b. DFM reports networks, IPs, and corresponding links.

UNIX-Based Processes

This section describes the part of the workflow that DFM performs for discovering components residing on UNIX-based machines. DFM defines the Operating System. For details, see the descriptions below of what DFM discovers for the following Operating Systems:

- "AIX" below
- "FreeBSD" on the next page
- "HPUX" on page 937
- "LINUX" on page 937
- "OpenBSD" on page 938
- "SunOs" on page 939
- "VMKernel" on page 939

Full command: `'uname -a'`

Note:

Before reporting the discovery, DFM makes the following verifications:

- If the destination IP is a virtual address, only the IP and host are reported.
- In the case of the ZLinux OS, when the host model is **s390x**, the host is defined by the IP and domain name.
- If the interface has an invalid MAC address, DFM does not report it.

AIX

DFM discovers:

1. The DHCP enabled network interfaces (**ps**).

   Full command: `'ps -aef | grep dhcpcd | grep -v grep'`

2. The network interfaces (MAC address, name, description) (**lsdev**, **entstat**)

Full command: `'lsdev -Cc adapter -S | egrep ^ent'`

3. The IPs (**ifconfig**).

   Full command: `'ifconfig -a inet'`

4. DFM defines the boot date, domain name, and default gateway in the same manner as for FreeBSD.

5. The model and vendor (**uname**).

   Full command: `'uname -M'`

6. The serial number (**lsattr**).

7. The OS version (**oslevel**).

FreeBSD

DFM discovers:

1. The DHCP enabled interfaces (**ps**).

   Full command: `'ps aux | grep dhclient | grep -v grep'`

2. The boot date (**uptime**).

3. The network interfaces (**name**, **MAC**, **IP**, **network mask**, **DHCPenabled flag**) and IPs (**ifconfig**).

   Full command: `'ifconfig -a'`

   The host is defined by the lowest MAC address among the network interfaces.

4. The OS version and host model (**uname**).

   Full command:

   `'uname -r'` for the version

   `'uname -m'` for the model

5. The domain name (**domainname**).

   Report only filtered name: `'(none)'`,`'localdomain'`

6. The BIOS UUID (**dmidecode**).

   Full command: `'dmidecode | grep UUID'`

7. The default gateway (**netstat**).

   Full command: `'netstat -r -n'`

HPUX

1. DFM discovers the network interfaces by one of the following methods:

   a. **nwmgr**

   b. **lanscan** (if **nwmgr** is unsuccessful)

2. DFM defines aliases (**netstat**) for the discovered interfaces.

   Full command: `'netstat -I'`

3. For each interface, DFM defines IPs (**ifconfig**).

4. DFM discovers the host model, boot date, OS version, serial number, and default gateway.

5. DFM discovers the OS flavor (**swlist**).

   Full command: `'swlist | grep -E "HPUX.*?OE"'`

LINUX

DFM discovers:

1. The DHCP enabled network interfaces (**ps**).

   Full command: `'ps aux | grep dhclient | grep -v grep'`

2. The IPs and network interfaces (MAC address, name, description) (**ifconfig**).

   Full command: `'ifconfig -a'`

3. The boot date, serial number (**dmidecode**), OS version, host model, domain name, and default gateway.

4. Information about HMC (Hardware Management Console) and its IPs (**lshmc**).

   Full command: `'lshmc -V'`

5. The BIOS UUID (**dmidecode**).

   Full command: `'dmidecode | grep UUID'`

6. The OS flavor (**redhat-release**).

   Full command: `'cat /etc/redhat-release'`

OpenBSD

DFM discovers:

1. The DHCP enabled interfaces (**ps**).

   Full command: `'ps aux | grep dhclient | grep -v grep'`

2. The boot date (**uptime**).

3. The network interfaces (**name**, **MAC**, **IP**, **network mask**, **DHCPenabled flag**) and IPs (**ifconfig**).

   Full command: `'ifconfig -a'`

   The host is defined by the lowest MAC address among the network interfaces.

4. The OS version and host model (**uname**).

   Full command:

   `'uname -r'` for the version

   `'uname -m'` for the model

5. The domain name (**domainname**).

   Report only filtered name: `'(none)','localdomain'`

6. The BIOS UUID (**dmidecode**).

   Full command: `'dmidecode | grep UUID'`

7. The default gateway (**netstat**).

   Full command:`'netstat -r -n'`

SunOs

DFM discovers:

1. The network interfaces (**netstat**)

   Full command:`'netstat -np'`

2. The IP addresses.

   Full command:`'ifconfig -a'`

3. The boot date, domain name, BIOS UUID, and default gateway.
4. The OS version and release (**uname**).

   Full command:`'uname -rv'`

5. The host model (**prtdiag**)
6. The manufacturer (**showrev**)
7. The serial number (**dmidecode**)

   Full command:`'dmidecode | grep UUID'`

VMKernel

DFM discovers:

1. The network interfaces (MAC address, name) and IPs (**esxcfg-vmknic**)

   Full command:`'esxcfg-vmknic -l'`

2. The boot date, OS version, and host model.
3. The domain name (**esxcfg-info**).

   Full command:`'esxcfg-info | grep Domain'`

4. The BIOS UUID (**esxcfg-info**).

   Full command: `'esxcfg-info | grep \'BIOS UUID\'`

5. The serial number (**esxcfg-info**).

   Full command: `'esxcfg-info -w | grep \'Serial Number\''`

6. The default gateway (**esxcfg-route**).

7. The OS flavor (**vmware**)

   Full command: `'vmware -v'`

Nexus Processes

This section describes the part of the workflow that DFM performs for discovering components residing on Nexus machines.

1. DFM gets the host name using the command **sh hostname**.

2. DFM gets version, build and feature information for the switch using the command **sh ver**.

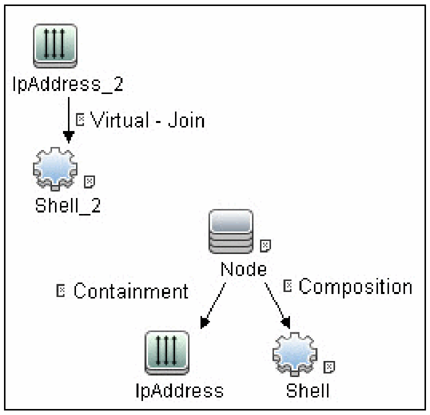
3. DFM gets dns server and local host data using the command **sh hosts**.

4. DFM get interface and configured IP information using the command **sh int**.

Trigger Query

- **Trigger CI**. The IP address.

- **Trigger TQL**. DFM uses this query to retrieve IPs that do not have Shell or have Shell with the same IP to reconnect.



- **Node conditions:**
  - IP Node

    Probe Name Is NOT null (IP Is Broadcast Equal false OR IP Is Broadcast Is NOT null)

Job Parameters

Parameter	Description
codepage	The discovered machine code page. **Default:** NA.
enableStamping	Determines whether or not **ud_unique_id** is stamped on the managed computer. **Default:** false.
language	The language of the discovered machine.
onlyStampingClient	Determines whether or not to only stamp the client machine. If set to false, both data-center and client machine are stamped. **Default:** true.
udaConnectionOrder	The position of UD in the order of protocol connections. Possible values are: **first**, **last**, and **none**. **Default:** last.
useAIXhwid	Whether to identify IBM AIX machines through their hardware ID. When set to **true** and used together with SNMP discovery, duplicate hosts may be created. If set to **false**, no AIX LPAR is discovered. **Default:** false.

Adapter

Triggered CI data:

- **ip_domain**. The domain of the IP address.
- **ip_address**. The IP address itself.

Discovered CITs

- Composition
- Containment
- DnsServer
- IPMP Group
- Interface
- IpAddress
- IpSubnet
- Membership
- NTCMD
- Node
- Parent
- Realization
- Remote Access Service
- Router
- Running Software
- SEA Adapter
- SNMP

- SSH
- Switch
- Telnet
- Terminal Server
- Unix
- Usage
- VAX
- Windows

Troubleshooting and Limitations

Troubleshooting

- **Problem**: When running the **Host Connection by Shell** job, the following error may be displayed:

  ```
  Error: Multiple connections to a server or shared resource by the same user,
  using more than one user name, are not allowed.
  ```

  Solution: This may be caused by one of the following NetBIOS protocol limitations:

  - The network share is considered to be in use even though it is not, that is, the session is frozen. In this case, try the following command:

    ```
    net use * /delete
    ```

  - The network share is in use by another user whose user name is bound to the local machine user name. In this case, you can reconfigure the remote machine security policy, or wait for the other user to finish working.

- **Problem**: If **HPCmd Commands Execution Context** is set to **User**, the HC by Shell job fails NTCMD discovery if the user's account does not have the right to **Log on as a service**.

  Solution: The user's account must have the right to **Log on as a service**. For details how to configure users with the right to **Log on as a service**, see http://technet.microsoft.com/en-us/library/cc739424(v=ws.10).aspx

Limitations

- **Limitation:** If an interface has a MAC address of 0, the job does not report that interface or the IP address assigned to it.
- **Limitation:** This discovery supports the reporting of PAE state only for Windows Operating systems.

Host Connection by SNMP Job

This subject includes the following sections:

Discovery Mechanism

1. DFM runs through the credentials defined for the protocol and tries to connect successfully through one of them. Whenever there is a successful connection, the discovery remembers the last successful credential and caches it for reuse the next time the discovery is run.

2. DFM executes an SNMP query and obtains the class name, vendor name, host OS name, host model, host version, and host release:

```
Using OIDs:
    SNMP MIB-2 System 1.3.6.1.2.1.1
SNMP MIB-2 Interfaces 1.3.6.1.2.1.20
3.
x3x.x3.x.xxxxxxxxxxx x
The vendor's authoritative identification of the network management subsystem
obtained from the system table.
```

3. DFM retrieves the host IP and mask:

```
Using OIDs:
    ipAdEntNetMask (1.3.6.1.2.1.4.20.1.3) for subnet mask
    ipAdEntBcastAddr (1.3.6.1.2.1.4.20.1.4) for the least-significant bit in
the IP broadcast address
    ipAdEntIfIndex (1.3.6.1.2.1.4.20.1.2) for the index value which uniquely
identifies the interface
```

4. DFM retrieves the network interface information:

```
OID (1.3.6.1.2.1.2.2.1) - an interface entry containing objects at
the subnetwork layer and below for a particular interface.
```

5. DFM retrieves the default gateway:

   ```
   Used OIDs:
       ipRouteDest (1.3.6.1.2.1.4.21.1.1) -
    for the destination IP address of this route
       ipRouteMask (1.3.6.1.2.1.4.21.1.11) -
    for the mask
       ipRouteDest (1.3.6.1.2.1.4.21.1.1) -
    for the destination IP address of this route
       ipRouteMetric1 (1.3.6.1.2.1.4.21.1.3) -
    for the primary routing metric for this route
       ipRouteNextHop (1.3.6.1.2.1.4.21.1.7) -
    for the IP address of the next hop of this route
   ```

6. DFM retrieves the serial number of the host. It will get the serial number from a public MIB, or failing that a private MIB. In both cases it retrieves the OID. This job supports a wide range of devices. However, should the serial number be available but DFM is unable to extract it, you should open a Support Case (ensuring you provide full details from MIB Walk) so we can add support in a future Content Pack or Update.

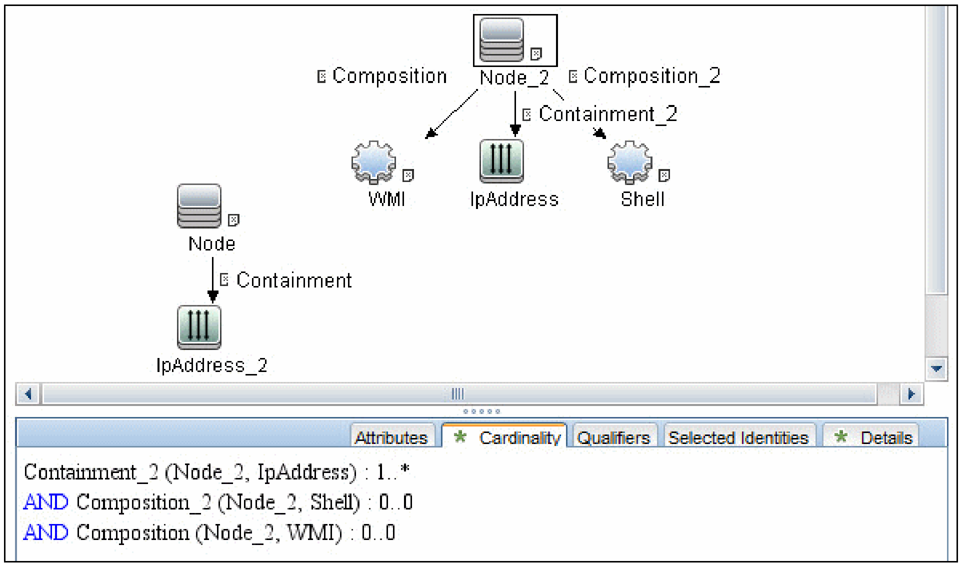
7. If possible, DFM retrieves remote management cards of the host. For example, HP iLO cards, or Dell DRAC cards.

   ```
   Get iLO cards from OID table 1.3.6.1.4.1.232.9.2.5.1
   ```

   ```
   Get DRAC cards from OID table 1.3.6.1.4.1.674.10892.1.1900.10
   ```

Trigger Query

- **Trigger CI**. The IP address.
- **Trigger TQL**This query enables the retrieval of IPs that are either (a) not connected to a Node by a Containment link; or (b) connected to a Node which has neither the Shell nor the WMI Agent.



Containment_2 (Node_2, IpAddress) : 1..*
AND Composition_2 (Node_2, Shell) : 0..0
AND Composition (Node_2, WMI) : 0..0

- **Node conditions**.
  - IP Node:

    ```
    NOT IP Lease Time equal Short
    AND NOT IP Probe Name Is null
    AND (IP Is Broadcast Equal false
    OR IP Is Broadcast Is null)
    ```

Job Parameters

None

Adapter

- Triggered CI data:
  - **ip_domain**. The domain of the IP address.
  - **ip_address**. The IP address itself.

Discovered CITs

- ATM Switch
- Composition

- Containment
- Firewall
- Interface
- IpAddress
- IpSubnet
- Load Balancer
- Mainframe
- Membership
- Net Device
- Net Printer
- Node
- Parent
- Remote Access Service
- Router
- SNMP
- Switch
- Terminal Server
- Unix
- VAX
- Windows

Troubleshooting and Limitations

- **Problem**: Following the run of the **Host Connection by SNMP** or **Host Networking by SNMP** jobs, many warning messages are displayed:

  ```
  Detected multiple updates in bulk - found attribute:
  'interface_description' on current CIT: 'interface'
  ```

  These messages can be safely ignored. To prevent the messages being displayed, you can change the **multipleUpdateIgnoreTypes** parameter in the **globalSettings.xml** file:

  ```
  <!--multipleUpdateIgnoreTypes
  - don't check multiple updates for the following types-->
  <property name="multipleUpdateIgnoreTypes">
  process,clientserver,node</property>
  ```

  Add the **interface** CIT to this list of CITs to be ignored.

- **Problem**: Host connection discovery uses the following workflow: **Host Connection by Shell**, then **Host Connection by WMI** and then **Host Connection by SNMP**. Therefore, if **Host Connection by Shell** is successful, neither of the following jobs complete. Also, if **Host Connection by WMI** is successful, **Host Connection by SNMP** does not complete.

  Solution: To skip this restriction, change the Trigger Query for these jobs:

  - Select **Host Connection by SNMP**.
  - Select the **Properties** tab.
  - Delete the Trigger Query **ip_with_snmp_or_without_host**.
  - Click the ➕ button in the **Trigger Query** section to create a new Trigger Query. The **Choose Discovery Query** dialog box appears.
  - Select **ip** from the list and click **OK**.

Host Connection by WMI Job

This subject includes the following sections:

Discovery Mechanism

1. DFM runs through the credentials defined for the WMI protocol and tries to connect successfully through one of them.

2. DFM performs a WMI query for `Win32_ComputerSystem` to retrieve the machine name.

   WMI query:

   ```
   select Name from Win32_ComputerSystem
   ```

   DFM performs a WMI query for **`Win32_NetworkAdapterConfiguration`** to retrieve the following interface information: IP addresses, MAC address, subnet IPs, description, and DHCP enabled attribute. DFM ignores local IPs in the interfaces.

   WMI query:

   ```
   'SELECT DnsHostName,IPAddress,MACAddress,IPSubnet,Description,
   DhcpEnabled FROM Win32_NetworkAdapterConfiguration
   WHERE MACAddress <> NULL'
   ```

3. DFM checks whether the destination IP address is a local IP address. If it is, DFM reports IPs and hosts only.

   If DFM cannot discover hosts by this manner, DFM tries to create a host defined by the lowest MAC address among the discovered network interfaces. If there is no interface to provide a valid MAC address, DFM defines the host by the destination IP address.

MAC addresses are used only in such interfaces that comply with the following rules:

- The interface has a valid MAC address.
- The interface does not belong to one of the following types: loopback, wireless, virtual, WAN miniport, RAS ASYNC, Bluetooth, FireWire, VPN, or IPv6 tunneling.
- The component is not the VMware interface, and the **ignoreVmwareInterfaces** option is not set to **1** in the **globalSettings.xml** configuration file.

4. DFM queries `Win32_OperatingSystem` to retrieve the host vendor, OS name, version, boot time, and installation type.

   WMI query:

   ```
   select Caption,Version,
   ServicePackMajorVersion,ServicePackMinorVersion,
   BuildNumber,Organization,RegisteredUser,TotalVisibleMemorySize,
   LastBootUpTime,OtherTypeDescription from Win32_OperatingSystem
   ```

5. DFM queries `Win32_IP4RouteTable` to retrieve the default gateway.

   WMI query:

   ```
   select NextHop, Metric1 from Win32_IP4RouteTable Where destination
   = '0.0.0.0' and mask = '0.0.0.0'
   ```

6. DFM queries `Win32_ComputerSystem` to retrieve the host manufacturer, the number of processors, host model, and OS domain.

   WMI query:

   ```
   select Manufacturer,NumberOfProcessors,Model,Domain from
   Win32_ComputerSystem
   ```

7. DFM retrieves the serial number by:

   - Querying `Win32_BaseBoard`.

     WMI query:

     ```
     SELECT SerialNumber FROM Win32_BaseBoard
     ```

   - Querying `Win32_SystemEnclosure`.

WMI query:

```
SELECT SerialNumber,SMBIOSAssetTag FROM Win32_SystemEnclosure
```

8. DFM queries `Win32_SystemEnclosure` to retrieve the system asset tag.

   WMI query:

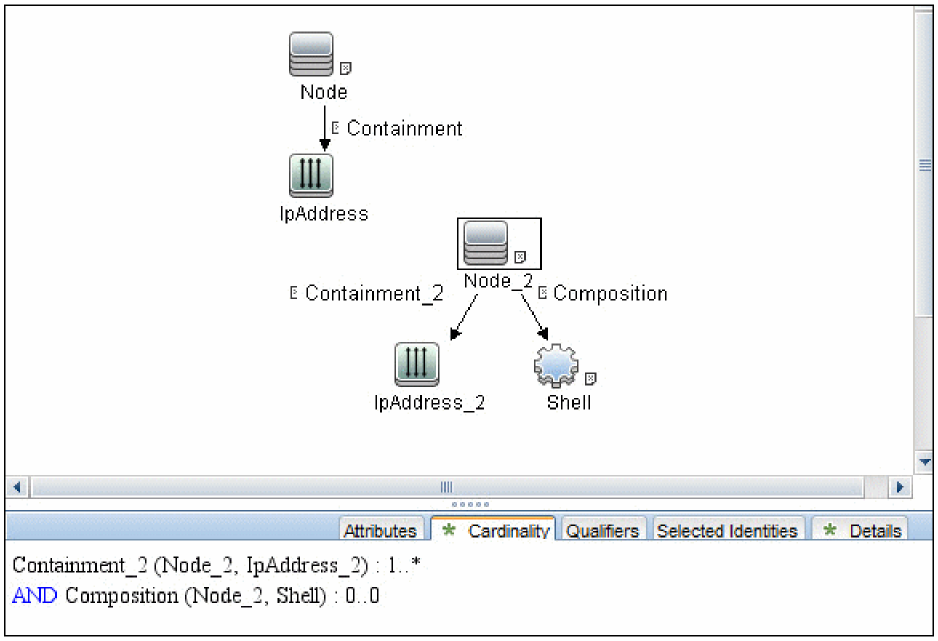
   ```
   SELECT SerialNumber,SMBIOSAssetTag FROM Win32_SystemEnclosure
   ```

9. If the connection is successful, DFM clears all errors and warnings that may have been generated in previous connection attempts, and returns the results.

10. If the connection is unsuccessful, DFM continues with the next WMI credential entry until all are tried.

Trigger Query

- **Trigger CI**. The IP address.
- **Trigger TQL**. This query enables the retrieval of IPs that are either (a) not connected to a Node by a Containment link; or (b) connected to a Node that does not have the Shell Agent.



Containment_2 (Node_2, IpAddress_2) : 1..*
AND Composition (Node_2, Shell) : 0..0

- **Node conditions**.

- IP Node:

  ```
  Probe Name Is NOT null
  (IP Is Broadcast Equal false OR IP Is Broadcast Is NOT null)
  ```

Job Parameters

None.

Adapter

Triggered CI data:

- **ip_domain**. The domain of the IP address.
- **ip_address**. The IP address itself.

Discovered CITs

- **Composition**
- **Containment**
- **Interface**
- **IpAddress**
- **IpSubnet**
- **Membership**
- **Node**
- **Parent**
- **WMI**

Troubleshooting and Limitations

- **Problem:** Host connection discovery uses the following workflow: **Host Connection by Shell**, then **Host Connection by WMI** and then **Host Connection by SNMP**. Therefore, if **Host Connection by Shell** is successful, neither of the following jobs complete. Also, if **Host Connection by WMI** is successful, **Host Connection by SNMP** does not complete.

Solution: To skip this restriction, change the Trigger Query for these jobs:

- Select **Host Connection by WMI**.
- Select the **Properties** tab.
- Delete the Trigger Query **ip_with_wmi_or_without_host**.
- Click the ✚ button in the **Trigger Query** section to create a new Trigger Query. The **Choose Discovery Query** dialog box appears.
- Select **ip** from the list and click **OK**.

Client Connection by SNMP Job

This subject includes the following sections:

Discovery Mechanism

1. DFM runs a ping sweep for client type IP ranges to get details of the active client machines.

2. DFM runs through the credentials defined for the protocol and tries to connect successfully through one of them. Whenever there is a successful connection, the discovery remembers the last successful credential and caches it for reuse the next time the discovery is run.

3. DFM executes an SNMP query and obtains the class name, vendor name, host OS name, host model, host version, and host release:

   ```
   Using OIDs:
       SNMP MIB-2 System 1.3.6.1.2.1.1
   SNMP MIB-2 Interfaces 1.3.6.1.2.1.20
   3.
   x3x.x3.x.xxxxxxxxxxx x
   The vendor's authoritative identification of the network management subsystem
   obtained from the system table.
   ```

4. DFM retrieves the host IP and mask:

   ```
   Using OIDs:
       ipAdEntNetMask (1.3.6.1.2.1.4.20.1.3) for subnet mask
       ipAdEntBcastAddr (1.3.6.1.2.1.4.20.1.4) for the least-significant bit in
   the IP broadcast address
       ipAdEntIfIndex (1.3.6.1.2.1.4.20.1.2) for the index value which uniquely
   identifies the interface
   ```

5. DFM retrieves the network interface information:

```
OID (1.3.6.1.2.1.2.2.1) - an interface entry containing objects at
the subnetwork layer and below for a particular interface.
```

6. DFM retrieves the default gateway:

```
Used OIDs:
    ipRouteDest (1.3.6.1.2.1.4.21.1.1) -
 for the destination IP address of this route
    ipRouteMask (1.3.6.1.2.1.4.21.1.11) -
 for the mask
    ipRouteDest (1.3.6.1.2.1.4.21.1.1) -
 for the destination IP address of this route
    ipRouteMetric1 (1.3.6.1.2.1.4.21.1.3) -
 for the primary routing metric for this route
    ipRouteNextHop (1.3.6.1.2.1.4.21.1.7) -
 for the IP address of the next hop of this route
```

7. DFM retrieves the serial number of the host. It will get the serial number from a public MIB, or failing that a private MIB. In both cases it retrieves the OID. This job supports a wide range of devices. However, should the serial number be available but DFM is unable to extract it, you should open a Support Case (ensuring you provide full details from MIB Walk) so we can add support in a future Content Pack or Update.

8. If possible, DFM retrieves remote management cards of the host. For example, HP iLO cards, or Dell DRAC cards.

```
Get iLO cards from OID table 1.3.6.1.4.1.232.9.2.5.1
```

```
Get DRAC cards from OID table 1.3.6.1.4.1.674.10892.1.1900.10
```

Trigger CI

Discovery Probe Gateway

Job Parameters

Parameter	Description
excludePatternsList	A list of wildcard patterns, separated by semicolons. IP addresses matching any of the patterns are skipped. The pattern may include numbers, dots, or the wildcards * (matches zero or more characters) or ? (matches exactly one character).

Parameter	Description
pingProtocol	A number representing the chosen ping protocol: 1 for ICMP, 2 for echo port, and 3 for both. **Default:** 1
range	Range of IP addresses to ping, separated by a semicolon.
retryDiscover	The number of times the job tries to ping. **Default:** 2
threadPoolSize	Number of threads in pool that performs port 7 echoing. **Default:** 10
timeoutDiscover	Ping timeout in milliseconds. **Default:** 3,000

Triggered CI Data

Name	Value
probeName	${SOURCE.name}

Discovered CITs

- ATM Switch
- Composition
- Containment
- Fibre Channel Switch
- Firewall
- Interface
- IpAddress
- IpSubnet
- LDOM Virtual Switch

- Load Balancer
- Mainframe CPC
- Marconi ATM Switch
- Membership
- Net Device
- Net Printer
- Node
- OpenVMS
- Parent
- Remote Access Service
- Router
- SNMP
- Switch
- Terminal Server
- Unix
- VMware Virtual Switch
- Windows

Chapter 69: DNS Zone Discovery

This chapter includes:

Overview

DNS Zone discovery retrieves the DNS Zone topology and records that belong to the zone. To transfer the zone, the machine performing the query should be included in a white list configured in the name server. This method requires a special DNS server configuration to permit Probe zone transfer.

The discovery mechanism triggers on a particular name server that records which zones should be reported, as follows:

1. Checks the **zoneList** parameter for the list of zones to transfer alias records.
2. Ignores zones with the name **arpa**, **localhost**, or '**.**' (root).
3. For each zone, transfers all records of type **CNAME** and **A** (second step). If the transfer fails, the zone is not reported.
4. Creates realization links.

For details, see "DNS Zone by nslookup Job" on page 964.

DNS Zone discovery is implemented in the following ways:

- The **DNS Zone by nslookup** job queries the DNS server for zone records from the Server itself. This method requires Shell access. For details, see "How to Discover DNS Zone by nslookup" on page 962
- The **DNS Zone by DNS** job queries the DNS server for zone records from the Data Flow Probe machine. This method requires a special DNS server configuration to permit Probe zone transfer. For details, see "How to Discover DNS Zone by DNS" on page 963

In the case where administrators do not want to add Shell access to DNS servers or read access to the configuration file, you can transfer zones specified in the mandatory **zone**List adapter parameter. For details, see "DNS Zone by nslookup Job" on page 964.

These implementations retrieve the same topology and have a common discovery mechanism that differs only in the client type (Server or Probe).

Note: The volume of retrieved topology data may be influenced by the parameters set for particular jobs.

Supported Versions

- Microsoft Windows 2000 Advanced Server or later
- UNIX-like OS BIND 9 name server

How to Discover DNS Zone by nslookup

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the following protocols:

   - SSH protocol
   - NTCMD protocol
   - Telnet protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Protocol parameters**

   - If some commands are configured to run with **sudo** on the target host, in the **Protocol Parameters** dialog box, fill in the following fields:
     - **Sudo paths**. Enter the full path to the **sudo** executable, together with the name of the executable. You can add more than one entry if executable files are placed in various places on the target operating systems.

       Example: `sudo,/usr/bin/sudo,/bin/sudo`
     - **Sudo commands**. Enter a list of the commands that are prefixed with the **sudo**.

       Example: `lspath,ifconfig`
   - Before activating discovery, confirm that the discovery user has all the required permissions to run the following command:

     ```
     cat <path to named config file and its include files>
     ```

     For details, see "Protocol Parameter Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job.

b. Run the **Host Connection by Shell** job.

c. Run the **Host Applications by Shell** job.

d. Run the **DNS Zone by nslookup** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover DNS Zone by DNS

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   Discovery is performed by the DNS protocol. To perform discovery, set up the following:

   - As all requests are performed from the Probe machine, this machine must be included in the list of servers that can transfer specified zone records. The administrator of the name server grants permissions to transfer the zone from the Probe machine.
   - Provide a list of zones that need to be transferred. For details, see "DNS Zone by nslookup Job" on the next page.

2. **Run the discovery**

   a. Run the **Range IPs by ICMP** job.

   b. Run the **TCP ports** job.

   c. Run the **DNS Zone by DNS** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Hosts by Shell using nslookup on DNS Server

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the following protocols:

   - NTCMD
   - SSH
   - Telnet
   - UDA

2. **Run the discovery**

   a. Run the **Range IPs by ICMP** job.

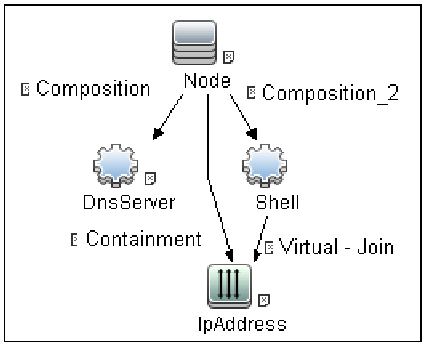
   b. Run the **Host Connection by Shell** job.

   c. Run the **Hosts by Shell using nslookup on DNS Server** job.

DNS Zone by nslookup Job

This job discovers the DNS Resource Record topology of DNS Zones by querying name the server using a remote shell.

Trigger Query



CI Attribute Conditions

CI	Attribute Value
Shell attributes	NOT Reference to the credentials dictionary entry is null

CI	Attribute Value
IP attributes	NOT IP Probe Name is null

Adapter Information

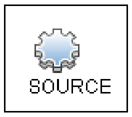
ID

DNS_Zone

Input CIT

Shell

Input Query



Triggered CI Data

Name	Value
connected_os_ credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}

Parameters

The adapter includes the following parameters:

Parameter	Description
reportBrokenAliases	If **true**, aliases that do not include a canonical resource are reported. This parameter is needed when an alias points to the address record or another alias record and this record cannot be found in the transferred data. The default value is **false**.

Parameter	Description
zoneList	Contains a comma separated list of zones to be transferred. This is an optional attribute. By default the zone list is not specified, so it is determined automatically.
includeOutscopeIPs	If IP address is out of probe range and parameter set to false IP address is not reported. The default value is “false”.

Used Script

- dns_zone_by_remote_shell.py

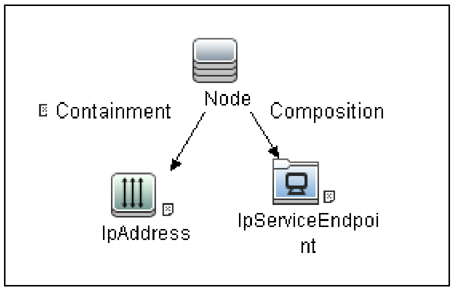
Discovered CITs

- Composition
- DnsRecord
- DnsZone
- IpAddress
- Realization

DNS Zone by DNS Job

This job discovers the DNS Resource Record topology of DNS Zone by querying the server name from the local shell (Probe) via the DNS protocol.

Trigger Query



CI Attribute Condition

CI	Attribute Value
IpServiceEndpoint	Name Equal dns AND NOT IP address is null

Adapter Information

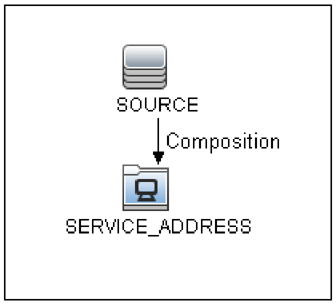
ID

DNS_Zone_by_DNS

Input CIT

Node

Input Query



Triggered CI Data

Name	Value
ip_address	${SERVICE_ADDRESS.bound_to_ip_address}

Used Scripts

- dns_zone_by_probe_shell.py

Discovered CITs

- Composition
- DnsRecord
- DnsZone
- IpAddress
- Realization

Parameters

Name	Description
includeOutscopeIPs	If the IP is out of the probe range and this parameter set to **false**, the IP is not reported. The default value is **false**.

Name	Description
reportBrokenAliases	If this parameter is set to **true**, aliases for which canonical resources do not exist are reported .
zoneList	Contains a comma separated list of zones to be transferred. This is a mandatory attribute.

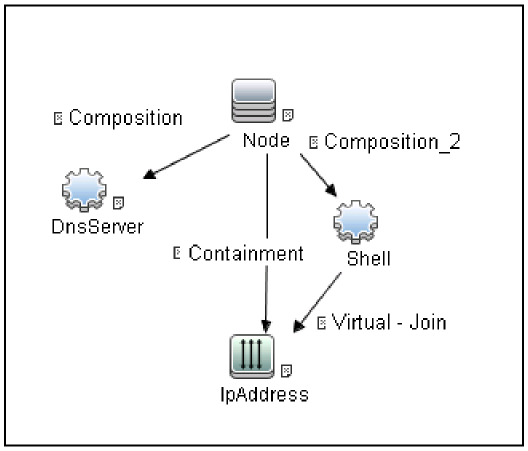
Hosts by Shell using nslookup on DNS Server Job

This job discovers hosts by querying all available DNS servers.

Adapter

ID: NSLOOKUP_on_DNS_Server

Trigger TQL



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
Shell	NOT Reference to the credentials dictionary entry Is null

Parameters

Name	Description
DNSServerDomain	The DNS Server Domain.
DNSServerName	The DNS Server Name.
discoverUnknownIPs	This flag determines whether to discover IPs that are out of the probe range.

Discovery Flow

1. Establish a connection with the DNS server.

2. Determine target DNS server and domain to transfer by checking the input parameters **DNSServerName** and **DNSServerDomain** respectively, and running the **nslookup** command to request the default server name and its domain.

3. List and process all IPv4 (A) DNS records.

4. Report hosts based on listed IP addresses.

NSLOOKUP on DNS Server Adapter

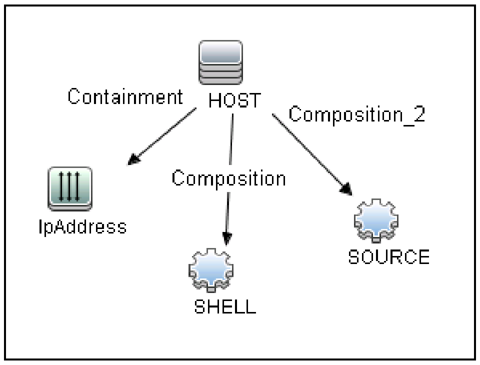
ID

NSLOOKUP_on_DNS_Server

Input CIT

Shell

Input TQL



Node Name	Condition
IpAddress	IpAddressType Equal IPv4
Shell	CI Type Equal ssh OR CI Type Equal uda

Triggered CI Data

Name	Value
Protocol	${SOURCE.root_class}
codepage	${SOURCE.codepage:NA}
credentialsId	${SOURCE.credentials_id}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
ip_address	${SOURCE.application_ip}
language	${SOURCE.language:NA}
sshKeyPath	${SHELL.ssh_keypath:NA}

Used Script

NSLOOKUP.py

Discovered CITs

- Windows
- Node
- IpAddress
- Net Printer
- Unix
- Containment

Global Configuration File

- globalSettings.xml

Parameters

Name	**Description**
DNSServerDomain	The DNS server domain.
DNSServerName	The DNS server name.
discoverUnknownIPs	If **true**, the job also discovers IPs that are out of probe range.

Discovery Mechanism – Windows

This section includes the following commands:

Query Windows Registry for Zone Information

Command

```
Reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS
Server\Zones"
```

Output

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS
Server\Zones\104.24.172.in-addr.arpa
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS
Server\Zones\foo.bar.net
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS
Server\Zones\od5.lohika.com
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS
Server\Zones\ucmdb-ex.dot
```

Mapping

CMD Output Attribute	CI Name	CI Attribute
Key name	DNS Zone	Name

List Root Domain to Transfer Resource Records

Zone resource records of type **CNAME** and **A** are transferred by listing the root domain of the zone in the **nslookup** command.

Command

```
echo ls -d <domain> | nslookup - <name server>
```

Output

```
Ns-2.od5.lohika.com. CNAME dc05-2.od5.lohika.com

od5.lohika.com. A  134.44.98.22
ftp.od5.lohika.com. CNAME  od5.lohika.com.
```

Mapping

CMD Output Attribute	CI Name	CI Attribute
First column	DNS Alias	Name
Third column	DNS Alias	Canonical name

Discovery Mechanism – UNIX-like

This section includes the following commands:

Parse Named Server Configuration File to Retrieve Zone Information

1. Try to find information about the named server configuration file in the command like the corresponding process.

   Command

   ```
   ps -ef | grep named | awk '{for(i=11; i < NF; i++) {printf("%s ", $i)}printf
   ("\n")}'
   ```

   Output

   ```
   /usr/sbin/named -t /var/lib/named -u
   ```

   Mapping

   The path specified for the **-t** option is the path to the configuration file.

2. If the path is recognized, the job tries to retrieve information about zones and include files to process. The default paths are **/etc/named.conf** and **/etc/namedb/named.conf**.

   Command

   ```
   cat <configuration file path> | awk '/zone|include/ {print}'
   ```

   Output

   ```
   zone "." in {
   zone "localhost" in {
   zone "od5.lohika.com" in {
   ```

   Mapping

CMD Output Attribute	CI Name	CI Attribute Display Name
Key name	DNS Zone	Name

List Root Domain to Transfer Resource Records

Zone resource records of type **CNAME** and **A** are transferred using the **dig** command and the **axfr** transfer type.

Command

```
dig @<server> <domain> axfr | awk '/(CNAME|A)/{print $1, "\t", $4, "\t", $5}'
```

Output

```
Ns-2.od5.lohika.com.  CNAME     dc05-2.od5.lohika.com
od5.lohika.com.         A         134.44.98.22
ftp.od5.lohika.com.   CNAME          od5.lohika.com.
```

Mapping

CMD Output Attribute	CI Name	CI Attribute Display Name
First column	DNS Alias	Name
Third column	DNS Alias	Canonical name

Glossary

- **CNAME record or Canonical Name record**

  A type of resource record in the Domain Name System (DNS) that specifies that the domain name is an alias of another canonical domain name.

- **Zone transfer**

  Listings of records contained in the zone.

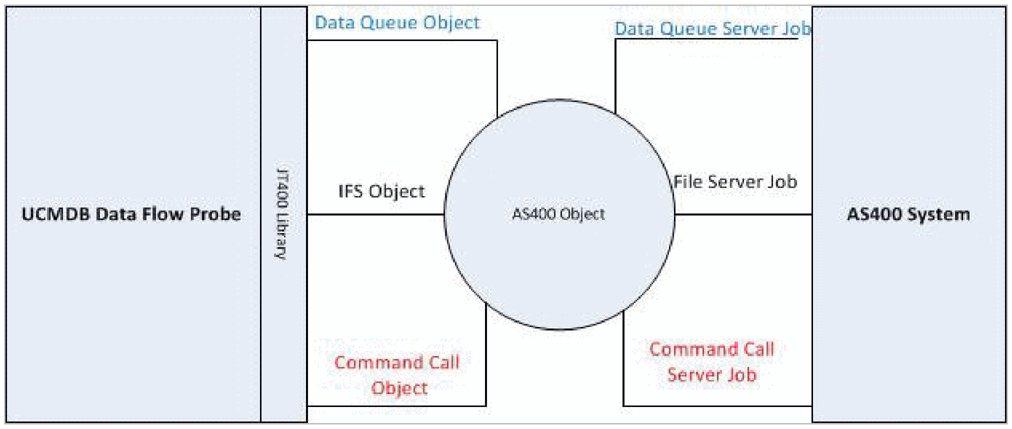
Chapter 70: AS400 Host Discovery

This chapter includes:

Overview

AS400 Host discovery is a simple host connection discovery for AS400 computers. The UCMDB Data Flow Probe uses an AS/400 object created by the IBM(R) jt400 library to access the AS400 system to retrieve host information.

A high-level architectural diagrams for this discovery solution is illustrated in the following image:



Supported Versions

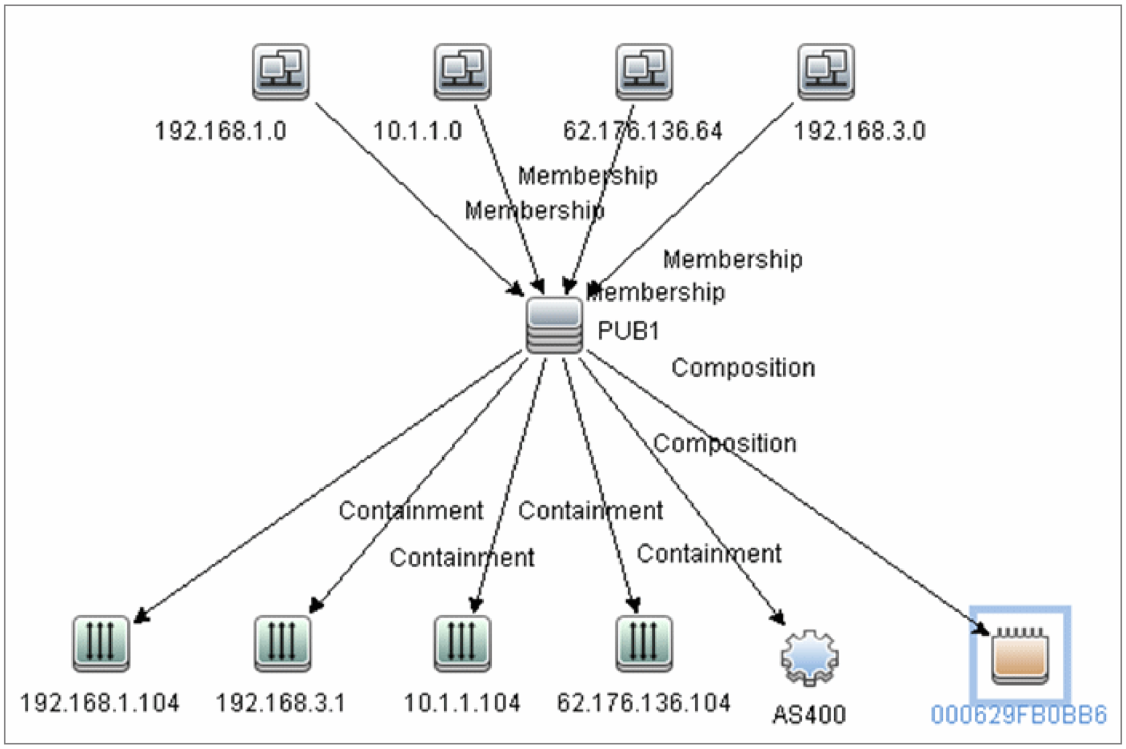
This discovery supports the following versions of AS400:

- V4R2M0
- V3R2M1
- V3R2M0
- V4R5M0
- V5R3
- V5R4M0
- V6R1

Topology

The following image displays the topology of the AS400 Host discovery with sample output:

Note: For a list of discovered CITs, see "Host Connection to AS400 Job" on page 982.



How to Discover AS400 Hosts

This task explains how to discover AS400 hosts and includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   This discovery uses the AS400 protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - IP Addresses and permissions**

   - Make sure that an IP ping sweep has been done on the ranges intended for AS400 host discovery.
   - Ensure that the user has the relevant permissions on the AS400 system to run the discovery.
     - *OBJMGT
     - *OBJEXIST
     - *ADD
     - *READ
     - *EXCLUDE
     - *EXECUTE
     - *CHANGE
     - *USE
     - *SHRNUP

3. **Run the discovery**

   Activate the **Host Connection to AS400** discovery job.

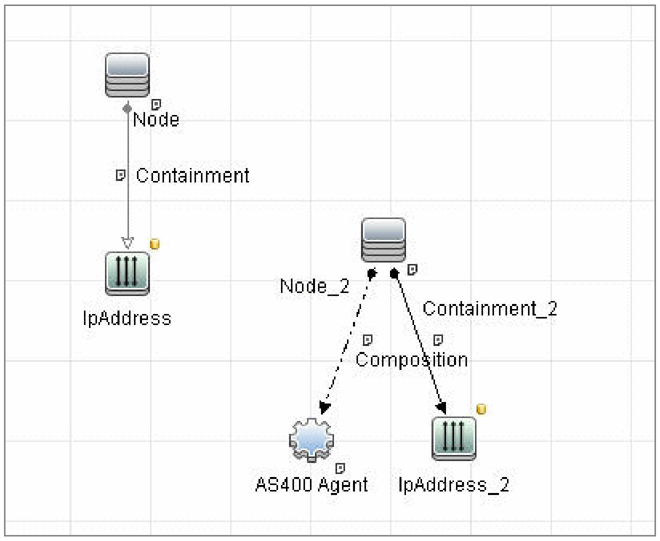
   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Host Connection to AS400 Job

This section includes details about the job.

Trigger Query

Trigger CI:ip_address



Discovered CITs

- AS400Agent
- Composition
- Containment
- Interface
- IpAddress
- IpSubnet
- Membership
- Node

- Parent

Note: To view the topology, see "Topology" on page 980.

Chapter 71: Host Connection by PowerShell Discovery

This chapter includes:

Overview

Windows PowerShell is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language built on top of, and integrated with, the .NET Framework. PowerShell provides full access to COM and WMI, enabling administrators to perform administrative tasks on both local and remote Windows systems.

In PowerShell, administrative tasks are generally performed by **cmdlets** (pronounced command-lets), which are specialized .NET classes implementing a particular operation. Sets of cmdlets may be combined together in scripts, executables (standalone applications), or by instantiating regular .NET classes (or WMI/COM Objects). These work by accessing data in different data repositories, like the file system or registry, which are made available to PowerShell via Windows PowerShell providers.

Supported Versions

This discovery supports PowerShell 2.0.

How to Discover Host Connection by PowerShell

The following sections describe the Host Connection by PowerShell discovery.

1. **Prerequisite - Set up protocol credentials**

   The Host Connection by PowerShell discovery solution is based on the PowerShell protocol.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Configure PowerShell**

   Before starting the discovery, ensure that PowerShell v2.0 is installed and configured on the Data Flow Probe machine. To access the installation files, see http://support.microsoft.com/kb/968929).

   a. Enable PowerShell remoting:

      - Launch PowerShell v 2.0 as an administrator.

      - Run the **Enable-PSRemoting** cmdlet. This starts the WinRM service and sets the startup type to Automatic, enables a firewall exception for WS-Management communications, and creates a listener to accept requests on any IP address.

        Note: To enable PowerShell remoting on all computers in your domain, in Domain Group Policy: Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > \WinRM Service, select **Allow automatic configuration of listeners**.

   b. To trust all hosts, run the following from the command line:

      ```
      Set-Item WSMan:\localhost\Client\TrustedHosts *
      ```

      To trust only restricted IP addresses, specify the addresses in place of the asterisk (*).

   c. Restart WinRM by running the following from the command line:

      ```
      restart-Service winrm
      ```

Note: By default, WinRM uses Kerberos for authentication. To configure WinRM for https, see http://support.microsoft.com/kb/2019527.

3. **Run the discovery**

   a. Run the **Range IPs by ICMP** job.

   b. Run the **Host Connection by PowerShell** job.

      For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Host Connection by PowerShell Job

This section includes details about the job.

Commands

This section describes each of the commands used by Host Connection by PowerShell discovery.

Command

```
Get-WmiObject -Query "SELECT BuildNumber, Caption, Version, csdversion,
lastBootUpTime, otherTypeDescription FROM Win32_OperatingSystem " | Format-List
BuildNumber, Caption, Version, csdversion, lastBootUpTime, otherTypeDescription
```

- **Output**

```
BuildNumber : 2600
Caption : Microsoft Windows XP Professional
Version : 5.1.2600
csdversion : Service Pack 3
lastBootUpTime : 20101108094626.357090+120
otherTypeDescription :
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
BuildNumber	Windows	Host Operating System Release
Caption(1)	Windows	Host Operating System
Version	Windows	Host Operating System Version
csdversion	Windows	Windows Service Pack
lastBootUpTime	Windows	Host Boot Time
Caption(2)	Windows	Host Operating System Installation Type

Command

```
Get-WmiObject -Query "SELECT Domain, Manufacturer, Model, Name FROM Win32_
ComputerSystem " | Format-List Domain, Manufacturer, Model, Name
```

- **Output**

```
Domain : od5.lohika.com
Manufacturer : INTEL_
Model : D946GZIS
Name : DDM-RND-SV
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
Domain	Windows	OS domain name
Manufacturer	Windows	PC manufacturer
Model	Windows	Host model
Name	Windows	Host name

Command

```
Get-WmiObject -Query "SELECT name, uuid FROM win32_ComputerSystemProduct " |
Format-List name, uuid
```

- **Output**

```
name :
uuid : EAB9B406-CE4F-DB11-9150-0013D4D0773D
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
Uuid	Windows	Host BIOS UUID
Name	Windows	Host model

Command

```
Get-WmiObject -Query "SELECT serialNumber FROM Win32_BIOS " | Format-List
serialNumber
```

- **Output**

```
serialNumber : BQJO749007TY
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
serialNumber	Windows	Host serial number

Command

```
Get-WmiObject -Query "SELECT serialNumber FROM Win32_SystemEnclosure " | Format-
List serialNumber
```

- **Output**

```
serialNumber : BQJO749007TY
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
serialNumber	Windows	Host serial number

Command

```
Get-WmiObject -Query "SELECT metric1, nextHop FROM Win32_IP4RouteTable WHERE
destination = '0.0.0.0' and mask = '0.0.0.0'" | Format-List metric1, nextHop
```

- **Output**

  ```
  metric1 : 20
  nextHop : 134.44.98.7
  ```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
nextHop where metric value is minimal	Windows	Default gateway

Command

```
Get-WmiObject -Query "SELECT dnsServerSearchOrder FROM Win32_
NetworkAdapterConfiguration WHERE domainDnsRegistrationEnabled <> NULL" | Format-
List dnsServerSearchOrder
```

- **Output**

  ```
  dnsServerSearchOrder : {16.110.135.51, 16.110.135.52}
  dnsServerSearchOrder : {134.44.98.21, 134.44.98.22}
  ```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs. Based on the IP addresses, incomplete hosts are created with the attached DNS Server application CI.

Command

```
Get-WmiObject -Query "SELECT WinsPrimaryServer, WinsSecondaryServer FROM Win32_
NetworkAdapterConfiguration WHERE WinsPrimaryServer <> NULL or WinsSecondaryServer
<> NULL" | Format-List WinsPrimaryServer, WinsSecondaryServer
```

- **Output**

```
WinsPrimaryServer : 16.232.7.246
WinsSecondaryServer : 16.236.105.246
```

- **Mapping**

The output of this command is used to fill in the attributes of the CIs. Based on the IP addresses, incomplete hosts are created with the attached WINS Server application CI.

Command

```
Get-WmiObject -Query "SELECT dhcpServer FROM Win32_NetworkAdapterConfiguration
WHERE dhcpServer <> NULL" | Format-List dhcpServer
```

- **Output**

```
dhcpServer : 134.44.98.22
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs. Based on the IP addresses, incomplete hosts are created with the attached DHCP Server application CI.

Command

```
Get-WmiObject -Query "SELECT Caption, Description, DhcpEnabled, IPAddress,
IPSubnet, MACAddress FROM Win32_NetworkAdapterConfiguration WHERE MACAddress <>
NULL" | Format-List Caption, Description, DhcpEnabled, IPAddress, IPSubnet,
MACAddress
```

- **Output**

```
Caption : [00000003] WAN Miniport (PPTP)
Description : WAN Miniport (PPTP)
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 50:50:54:50:30:30
Caption : [00000004] WAN Miniport (PPPOE)
Description : WAN Miniport (PPPOE)
DhcpEnabled : False
IPAddress :
IPSubnet :
 MACAddress : 33:50:6F:45:30:30
Caption : [00393219] WAN Miniport (IP)
Description : WAN (PPP/SLIP) Interface
```

```
DhcpEnabled : False
IPAddress : {16.213.65.117}
IPSubnet : {255.255.255.255}
MACAddress : 00:53:45:00:00:00
Caption : [00000007] Packet Scheduler Miniport
Description : Packet Scheduler Miniport
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 4A:6F:20:52:41:53
Caption : [00000008] Intel(R) PRO/100 VE Network Connection
Description : Intel(R) PRO/100 VE Network Connection - Teefer2 Miniport
DhcpEnabled : True
IPAddress : {134.44.99.108}
IPSubnet : {255.255.252.0}
MACAddress : 00:16:76:BE:7E:DD
Caption : [00000009] Packet Scheduler Miniport
Description : Packet Scheduler Miniport
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 00:16:76:BE:7E:DD
Caption : [00000013] Teefer2 Miniport
Description : Teefer2 Miniport
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 00:16:76:BE:7E:DD
Caption : [00000014] Teefer2 Miniport
Description : Teefer2 Miniport
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 4A:6F:20:52:41:53
```

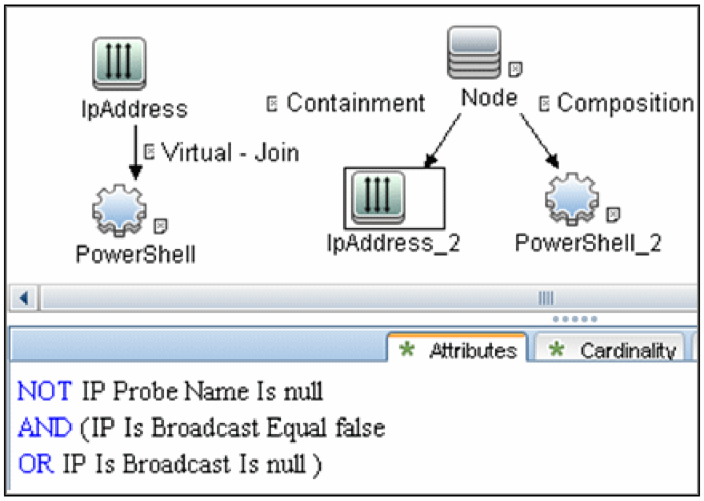
- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

Command Output Attribute	CI Type	CI Attribute
Description	Network Interface	Interface description
DhcpEnabled	Network Interface	DHCP Enabled
IPAddress	IP	IP address

Command Output Attribute	CI Type	CI Attribute
IPSubnet	IP	IP Network Address
MACAddress	Network Interface	Interface MAC Address

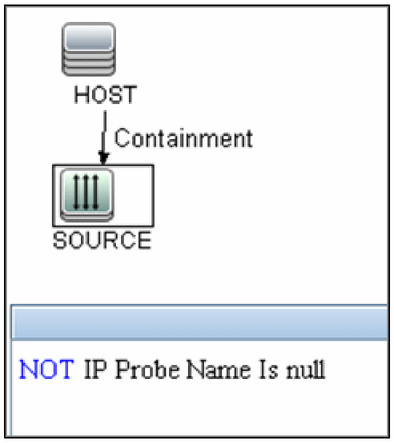
Trigger Query



NOT IP Probe Name Is null
AND (IP Is Broadcast Equal false
OR IP Is Broadcast Is null)

Adapter

- **Input query:**



NOT IP Probe Name Is null

- **Used Scripts:**

- Host_connection_by_powershell.py
- Host_win.py
- Host_win_shell.py
- Host_win_wmi.py
- Networking_win.py
- Networking_win_shell.py
- Networking_win_wmi.py

- **Triggered CI Data:**

Name	Value
host_cmdbid	${HOST.root_id:NA}
host_key	${HOST.host_key:NA}
ip_address	${SOURCE.ip_address}
ip_domain	${SOURCE.ip_domain}
mac_addrs	${NA}

Discovered CITs

- Composition
- Containment
- DnsServer
- Interface
- IpAddress
- IpSubnet
- Membership

- Node
- Parent
- PowerShell
- RunningSoftware
- Terminal Server
- Windows

Created/Changed Entities

Entity Name	Entity Type	Entity Description
powershell.xml	CIT	Represents the PowerShell protocol
Host Connection by Powershell.xml	Job	Main Job
Powershell_host_connection.xml	Adapter	Job adapter
Host_connection_by_powershell.py	Script	Discovery script
Host_win.py	Script	Discovery script
Host_win_shell.py	Script	Discovery script
Networking_win.py	Script	Discovery script
Networking_win_shell.py	Script	Discovery script
Networking_win_wmi.py	Script	Discovery script
Host_win_wmi.py	Script	Discovery script

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Host Connection by PowerShell Discovery.

Access Denied Error Message

The following error message may appear while trying to discover Windows 2008 SP2 destination by PowerShell protocol:

- *Connecting to remote server failed with the following error message: Access is denied. For more information, see the about_Remote_Troubleshooting Help topic.*

This appears if the user attempting to discover the destination host is not a local Administrator user. (It does not matter if the user is a member of the Administrators group.)

The solution requires additional configuration of PowerShell.

The **LocalAccountTokenPolicy** key should be changed to allow users from the Administrator group to connect remotely with Administrator privileges. Run the following command in PowerShell on the discovered host:

- Set-ItemProperty -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System -Name LocalAccountTokenFilterPolicy -Value 1 -Type DWord

For details of this special case, see "HOW TO ENABLE REMOTING FOR ADMINISTRATORS IN OTHER DOMAINS" at http://technet.microsoft.com/en-us/library/dd347642.aspx.

Chapter 72: Layer2 Discovery

This chapter includes:

Overview

The Layer2 package discovers the Layer2 topology that includes the switches tree topology (the backbone links between the switches) and also the end user connections to the switch-ports (the Layer2 CIs between a switch and a host).

The package can discover information from the following resources:

- **Forwarding Database (FDB)**
  Depending on device type, the FDB is found in the BRIDGE-MIB, QBRIDGE MIB, or STATISTICS-MIB.
- **Cisco Discovery Protocol MIB (CDP-MIB)**
  This MIB contains information collected by CDP about directly connected Cisco network devices.
- **Link Layer Discovery Protocol MIB (LLDP-MIB)**
  This MIB contains information collected by LLDP about directly connected network devices.

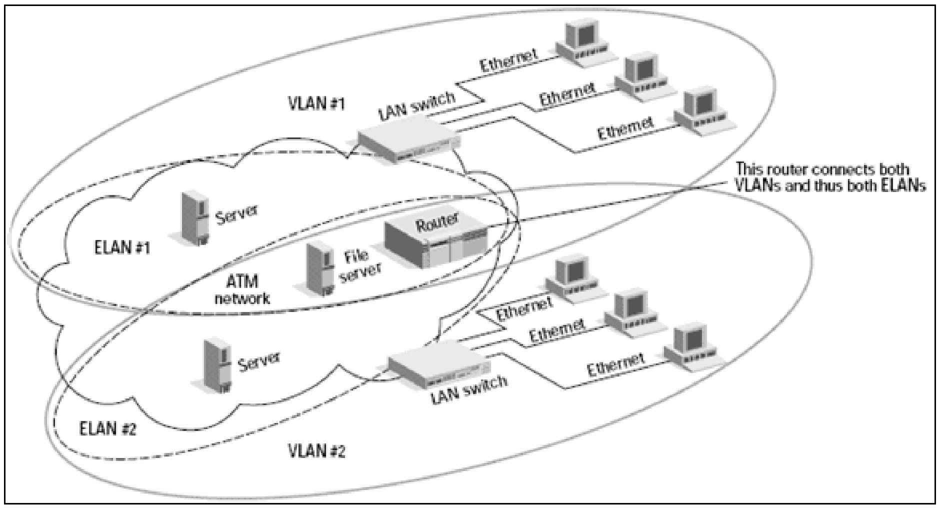
The **Layer2 Topology Bridge-based by SNMP** and **Layer2 Topology VLAN-based by SNMP** jobs select Layer2 connections information from the forwarding database, create the Layer2 CIs between a switch and an endpoint host, and put information about switch to switch Layer2 connections into files on the probe's file system.

The **Process Layer2 Saved Files** job selects data from the probe's file system (created by the **Layer2 Topology Bridge-based by SNMP** and **Layer2 Topology VLAN-based by SNMP** jobs) and creates switch to switch Layer2 connections.

The **Layer2 Topology CDP-LLDP based by SNMP** job selects Layer2 connections information from CDP or LLDP MIBs, and reports Layer2 connections from the discovered device to directly connected network devices.

The Layer2 package is based on the SNMP protocol.

The following image illustrates a router connecting overlapping VLANs/ELANs:



Supported Devices

This discovery supports devices by the following:

- 3Com
- Cisco
- H3C
- HP ProCurve

How to Discover Layer2 Objects

This task describes how to discover Layer2 objects.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   The SNMP protocol is required to discover Layer2 objects. When defining the SNMP protocol credentials, have available the Port and Community authentication parameters.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisite - Other**

   - Make sure that there is SNMP access to all switches in the environment to be discovered. This is a key requirement for fully discovering the Layer2 topology.

3. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Activate the jobs in the following order:

   a. Activate the **Host Connection by SNMP** job. This job saves SNMP CIs to the CMDB.

      Note: Layer2 discovery is based on the connection jobs for the following reasons:

      - The Layer2 connectivity between the switch-port to the host is based on the host MAC address. These MAC addresses are discovered by the network connection jobs (Host Interfaces).

      - The trigger of the Layer2 job is dependent on the type of the discovered switch. The switch class and type is discovered by the Host Networking by SNMP job for the Layer2 module.

   b. Activate the **Host Networking by SNMP** job. This job discovers host networking topology using SNMP route and system tables. You should run this job on all SNMP agents on the switches that were discovered in the environment. The to-be discovered Layer2 link names are dependent on this discovery. (Layer2 CIs names are the same as the relevant interface name and interface description on the destination network interface adapter which we are discovering.)

c. Activate the **VLANS by SNMP** job.

   The trigger for this job is the **snmp_of_catalyst_switch** query. The Switch CIT is either:

   - an SNMP object
   - an SNMP agent that is connected to a switch

   The `SNMP_Net_Dis_Catalyst_Vlans.py` script retrieves the VLAN, ELAN name, and VLAN number per ELAN tables.

d. Activate the **VLAN ports by SNMP** job.

   The trigger for this job is the **catalyst_vlan** query. This is a VLAN object that has a connection to:

   - a switch with an SNMP object
   - a switch

   The trigger is placed on the VLAN object instead of on the SNMP itself because the VLAN object must be authenticated with a special community string (and not with the regular community string that was discovered on the SNMP object on the discovered switch). This community string should hold the value `<COMMUNITY>@<VLAN NUMBER>`. For example, if the community string is **public** and the discovered VLAN number is **16**, the community string is **public@16**.

   Note: Because community string indexing can only be used for Cisco network devices, this job only works for Cisco equipment.

   The `SNMP_Net_Dis_VMS_catalyst.py` script retrieves the Base MAC table and Port number If Index table.

   For details on the SNMP protocol parameters, see SNMP Protocol in the *HP Universal CMDB Data Flow Management Guide*.

e. Activate the **Layer2 Topology Bridge-based by SNMP** job.

   The trigger for this job is the **catalyst_bridge_no_vlan** query. This is a Bridge object that has a connection to:

- a switch with an SNMP object
- a switch

Both this job (**Layer2 Topology Bridge-based by SNMP**) and the following job (**Layer2 Topology VLAN-based by SNMP**) use the `bridgePortDisc.py` script. The difference between the jobs in this script is as follows:

For Cisco network devices.

- **Layer2 Topology Bridge-based by SNMP** uses the regular SNMP community authentication. The job is triggered on the Bridge only when the discovered switch has no VLANS.
- **Layer2 Topology VLAN-based by SNMP** is triggered on each one of the VLANs discovered on the switch. This job uses the relevant special community authentication, as explained in "Activate the VLAN ports by SNMP job." on the previous page, based on the triggered VLAN number.

For other network devices.

- **Layer2 Topology Bridge-based by SNMP** uses the BRIDGE-MIB to discover Layer2 connections.
- **Layer2 Topology VLAN-based by SNMP** uses QBRIDGE-MIB or STATISTICS-MIB to discover Layer2 connections.

Note:

- The Layer2 Topology Bridge-based by SNMP job discovers Layer2 connections per default VLAN. (The default VLAN is #1.) The Layer2 Topology VLAN-based by SNMP job discovers all Layer2 connections for all VLANs, including the default one. Therefore:
  - If you want to discover Layer2 connections for the default VLAN only, you do not need to run the VLANs by SNMP and LVAN ports by SNMP jobs.
  - If you execute the VLANs by SNMP job, and it reports any VLAN CIs connected to the network device, the Layer2 Topology Bridge-based by SNMP job does not discover Layer2 connections because the device is excluded from the job's trigger TQL query. To discover Layer2 connections on a device with discovered VLANs, you must use the Layer2 Topology VLAN-based by SNMP job.

- If you dispatch the Bridge-based Layer2 job on the bridge of a switch that holds VLANs, only the default VLAN Layer2 topology is discovered.
- The Layer2 Topology Bridge-based/VLAN-based by SNMP jobs only report switches to host Layer2 connections. To report switch to switch, you must also run the Process Layer2 Saved Files job.

f. Activate the **Layer2 Topology VLAN-based by SNMP** job.

   The trigger for this job is the **catalyst_vlan_with_bridge** query. This is a VLAN object with a value in its `bridge_mac` attribute. It should also have a connection to either:

   - a switch with an SNMP object
   - a switch

   For details on the **bridgePortDisc.py** script, see "Activate the Layer2 Topology Bridge-based by SNMP job." on page 1001.

How to Discover Layer2 Connections Using Saved Files

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   The SNMP protocol is required to discover Layer2 objects. When defining the SNMP protocol credentials, have available the Port and Community authentication parameters.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Run the jobs in the following order:

a. Run the **Range IPs by ICMP** job to discover the target IPs.

b. Run the **Host Connection by SNMP** job to discover the target host and connectivity to it.

c. Run either or both of the **Layer2 Topology Bridge-based by SNMP** and **Layer2 Topology VLAN-based by SNMP** jobs, according to your environment.

d. After all the preceding jobs have completed, run the **Process Layer2 Saved Files** job.

How to Discover Layer2 Connections Using CDP or LLDP MIB

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   The SNMP protocol is required to discover Layer2 objects. When defining the SNMP protocol credentials, have available the Port and Community authentication parameters.

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Run the discovery**

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

   Run the jobs in the following order:

   a. Run the **Range IPs by ICMP** job to discover the target IPs.

   b. Run the **Host Connection by SNMP** job to discover the target host and connectivity to it.

   c. Run the **Layer2 Topology CDP-LLDP based by SNMP** job.

How to Discover Layer2 Topology by Shell

To discover Layer2 Topology by Shell, run the following jobs:

1. Range IP by ICMP
2. Host Connection by Shell
3. Layer2 Topology by Shell

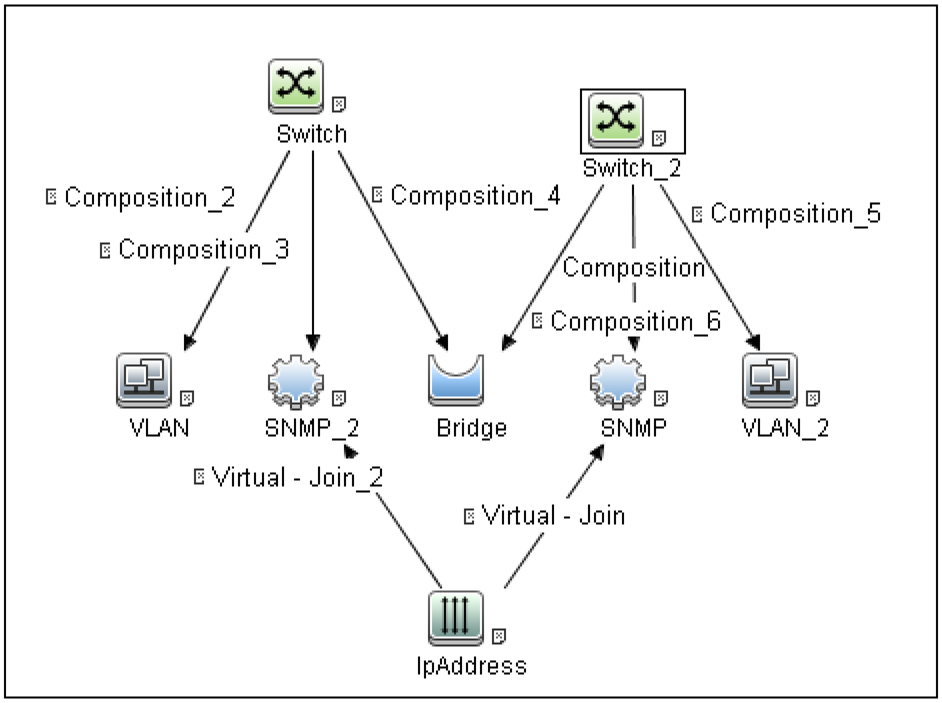
Layer2 Topology Bridge-based by SNMP Job

This section includes details about the job.

Adapter

This job uses the L2 Bridge by SNMP adapter.

Trigger TQL Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
IpAddress_2	NOT IP Probe Name Is null
SNMP	NOT Reference to the credentials dictionary entry is null
SNMP_2	NOT Reference to the credentials dictionary entry is null

Discovery Flow

This discovery includes the following parts:

- Connection to the destination machine.
- Discovery of VLAN related data.

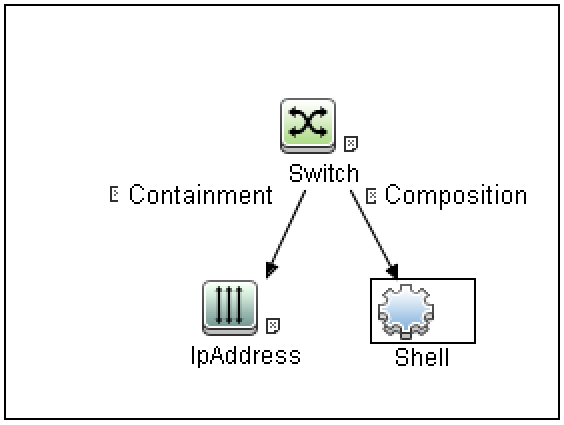
Layer2 Topology by Shell Job

This job reports Layer2 related data for Nexus 1000 Switches.

Adapter

ID: `Layer2 Topology by Shell`

Trigger TQL



Parameters

None

Prerequisites

- Set up SSH protocol credentials. For more information on this, see the section explaining SSH protocol credentials in *HP UCMDB Universal Discovery Content Guide - Supported Content*.

Discovery Flow

The discovery flow for the Layer2 Topology by Shell job is as follows:

1. Get interface details using the command **sh int**.
2. Get configured vlans and ports using the command **sh vlan all-ports**.
3. Get the list of connected devices using the command **sh cdp neighbours detail**.

Layer2 Topology CDP-LLDP based by SNMP Job

This section includes details about the job.

Adapter

This job uses the CDP/LLDP neighbors layer 2 devices by SNMP adapter.

Trigger TQL Query

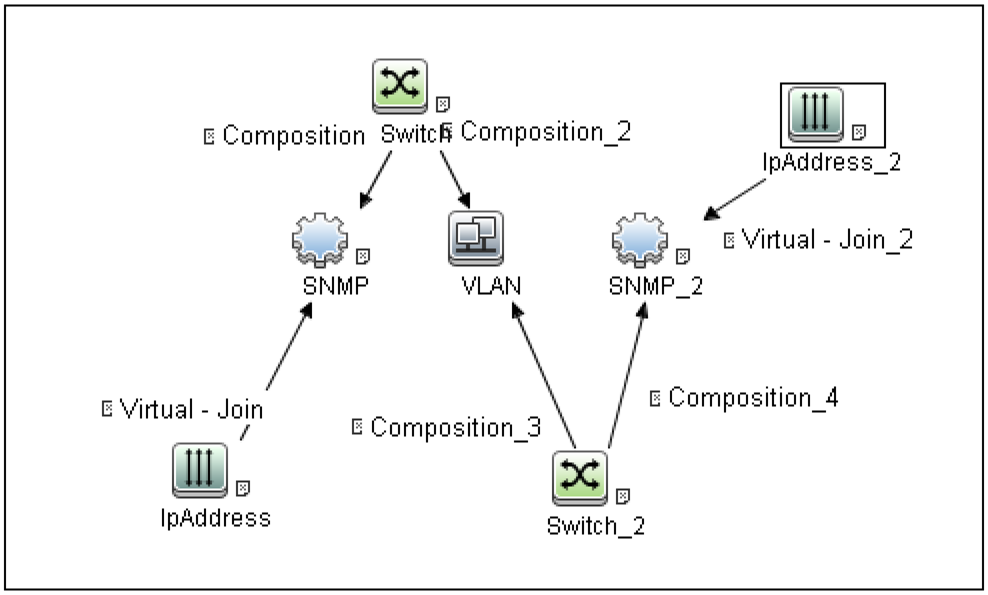
Node Name	Condition
Node	CI Type Equal switch OR CI Type Equal switchrouter OR NodeRole Contains switch OR NodeRole Contains router
SNMP	NOT Reference to the credentials dictionary entry is null

Layer2 Topology VLAN-based by SNMP Job

Adapter

This job uses the L2 Vlan by SNMP adapter.

Trigger TQL Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
IpAddress_2	NOT IP Probe Name Is null
SNMP	NOT Reference to the credentials dictionary entry is null
SNMP_2	NOT Reference to the credentials dictionary entry is null
VLAN	NOT VLAN Bridge MAC Is null

Discovery Flow

This discovery includes the following parts:

- Connection to the destination machine.
- Discovery of VLAN related data.

Process Layer2 Saved Files Job

This section includes details about the job.

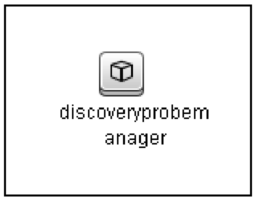
Discovery Flow

This job finds any connection between the MAC of one switch interface and the MAC of another switch interface. Where there is such a connection, the discovery creates a Layer2 Connection.

Adapter

This job uses the Process Layer2 Collected Files adapter.

Trigger TQL Query



Merge VLANs by Ports Job

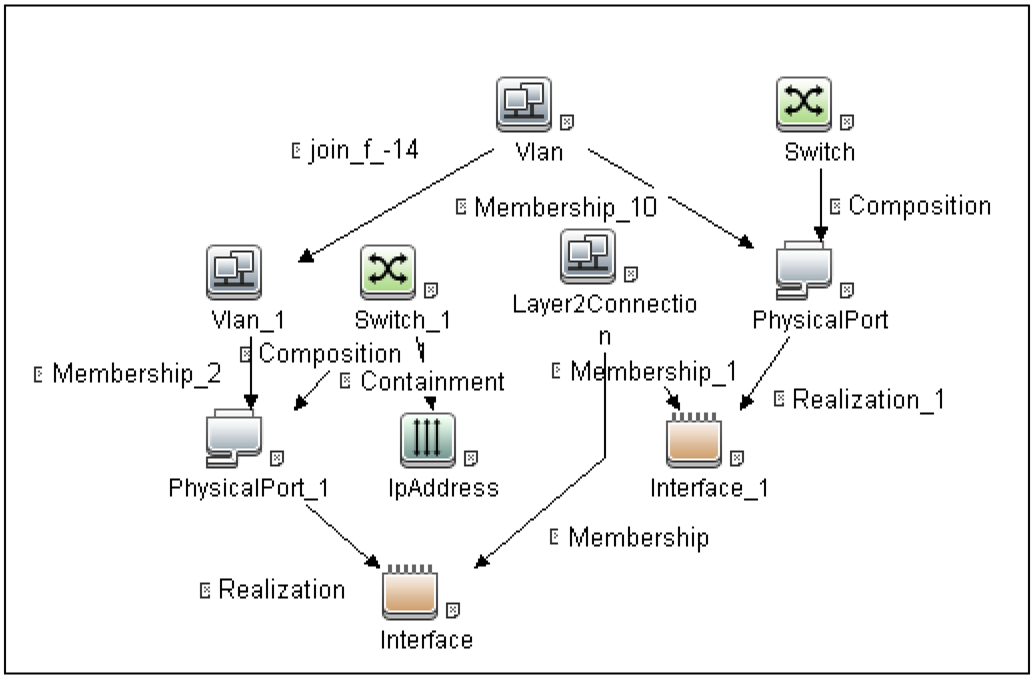
The functionality of this job is similar to that of enrichment or reconciliation. It works only with data which is already inside UCMDB, and merges VLANs where the topology is as follows:

1. The ports that are related to a VLAN are connected by a Layer2 Connection; and
2. The VLAN id is the same.

Adapter

This job uses the Merge VLANs adapter

Trigger TQL Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null

Discovery Flow

This discovery includes the following parts:

- Connection to the destination machine.
- Discovery of VLAN related data.

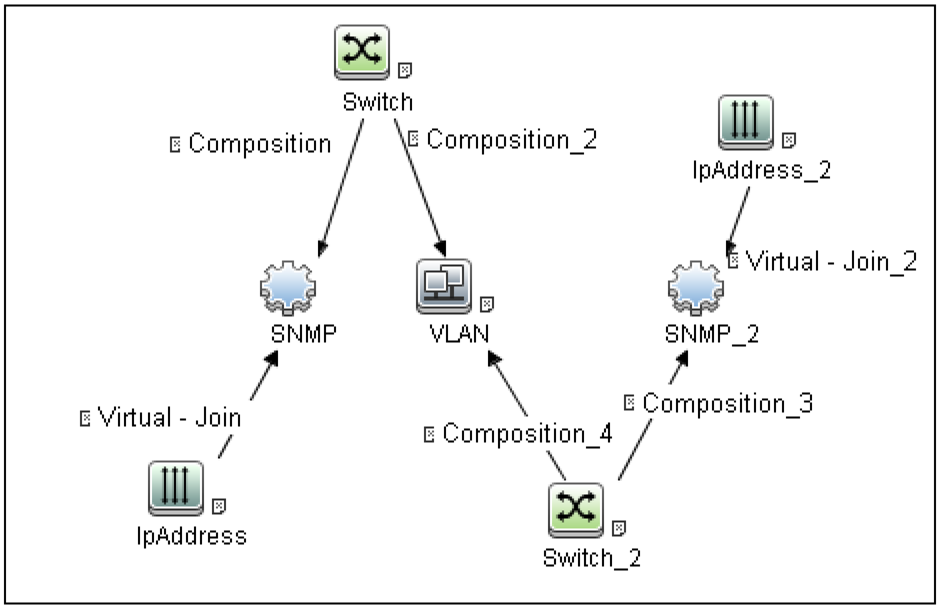
VLAN ports by SNMP Job

This section includes details about the job.

Adapter

This job uses the VMS Catalyst by SNMP adapter.

Trigger TQL Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null
IpAddress_2	NOT IP Probe Name Is null
SNMP	NOT Reference to the credentials dictionary entry is null
SNMP_2	(SNMP Description Like %atalyst% OR SNMP Description Like ignore case %cisco%) AND NOT Reference to the credentials dictionary entry Is null
Switch	DiscoveredOsName Like %atalyst% OR DiscoveredModel Like %atalyst% OR DiscoveredOsName Like ignore case %cisco% OR DiscoveredModel Like ignore case %cisco%

Discovery Flow

This discovery includes the following parts:

- Connection to the destination machine.
- Discovery of VLAN related data.

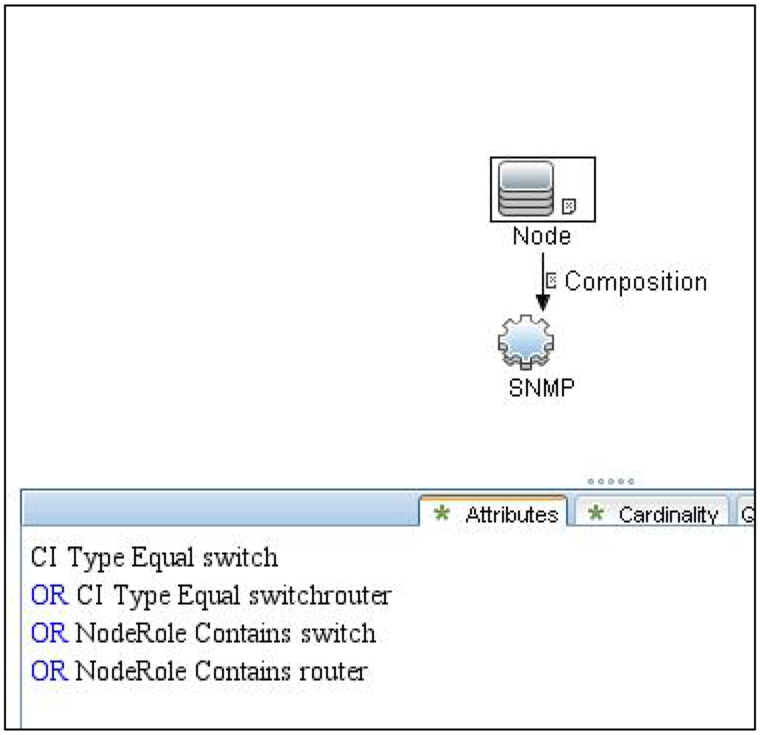
VLANS by SNMP Job

This section includes details about the job.

Adapter

This job uses the Catalist VLANS by SNMP adapter.

Trigger TQL Query



CI Type Equal switch
OR CI Type Equal switchrouter
OR NodeRole Contains switch
OR NodeRole Contains router

Discovery Flow

This discovery includes the following parts:

- Connection to the destination machine.
- Discovery of VLAN related data.

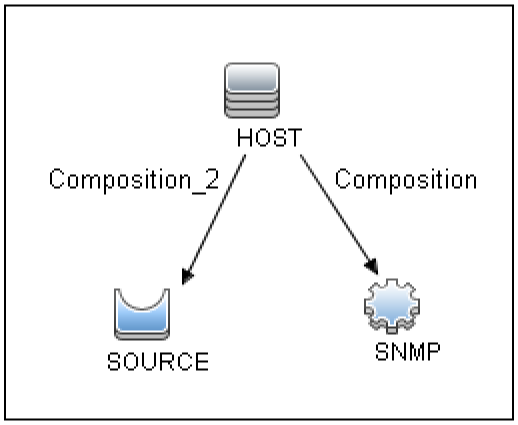
L2 Bridge by SNMP Adapter

This section includes details about the adapter.

Input CIT

Bridge

Input TQL Query



Triggered CI Data

Name	Value
bridgeId	${SOURCE.bridge_basemacaddr}
credentialsId	${SNMP.credentials_id}
hostId	${HOST.root_id}
ip_address	${SNMP.application_ip}

Used Scripts

- bridgePortDisc.py
- networking_win.py
- SNMP_Networking_Utils.py

Discovered CITs

- Bridge
- Composition
- Interface
- Layer2Connection
- Membership
- Node
- PhysicalPort
- Realization

Layer2 Topology by Shell Adapter

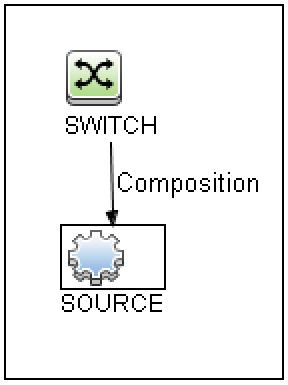
ID

```
Layer2 Topology by Shell
```

Input CIT

```
Shell
```

Input TQL



Triggered CI Data

Name	Value
credentialsId	SOURCE.credentials_id
hostId	SWITCH.root_id
ip_address	SOURCE.appliaction_ip
ip_domain	${NA}
Protocol	SOURCE:root_class

Used Scripts

- layer2.py
- layer2_shell_discoverer.py
- switch_layer2_by_shell.py
- TTY_Connection_Utils.py

Discovered CITs

- Composition
- Interface
- Layer2Connection
- Membership (layer2_connection,interface)
- Node
- Realization
- Switch
- PhysicalPort

Parameters

None

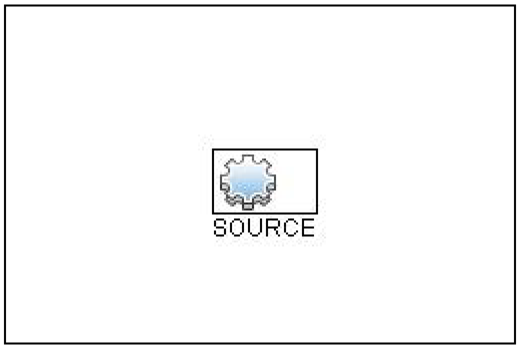
CDP/LLDP Neighbors Layer 2 Devices by SNMP Adapter

This section includes details about the adapter.

Input CIT

SNMP

Input TQL Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}

Used Scripts

SNMP_CDP_LLDP.py

Discovered CITs

- Composition
- Containment
- Interface

- IpAddress
- Layer2Connection
- Membership
- Node

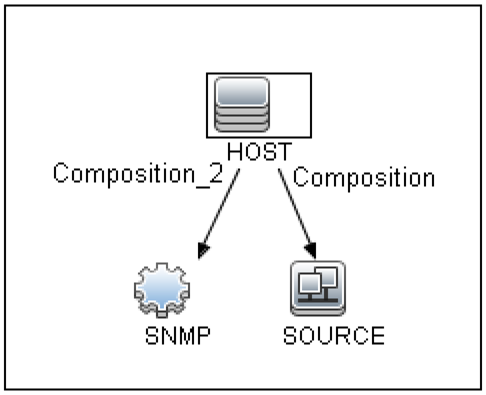
L2 Vlan by SNMP Adapter

This section includes details about the adapter.

Input CIT

Vlan

Input TQL Query



Triggered CI Data

Name	Value
bridgeId	${SOURCE.vlan_bridgemac}
credentialsId	${SOURCE.credentials_id}
hostId	${SOURCE.application_ip_domain}
ip_address	${HOST.root_id}
snmpCommunityPostfix	${SOURCE.vlan_id}

Used Scripts

- bridgePortDisc.py
- networking_win.py
- SNMP_Networking_Utils.py

Discovered CITs

- Bridge
- Composition
- Interface
- Layer2Connection
- Membership
- Node
- PhysicalPort
- Realization

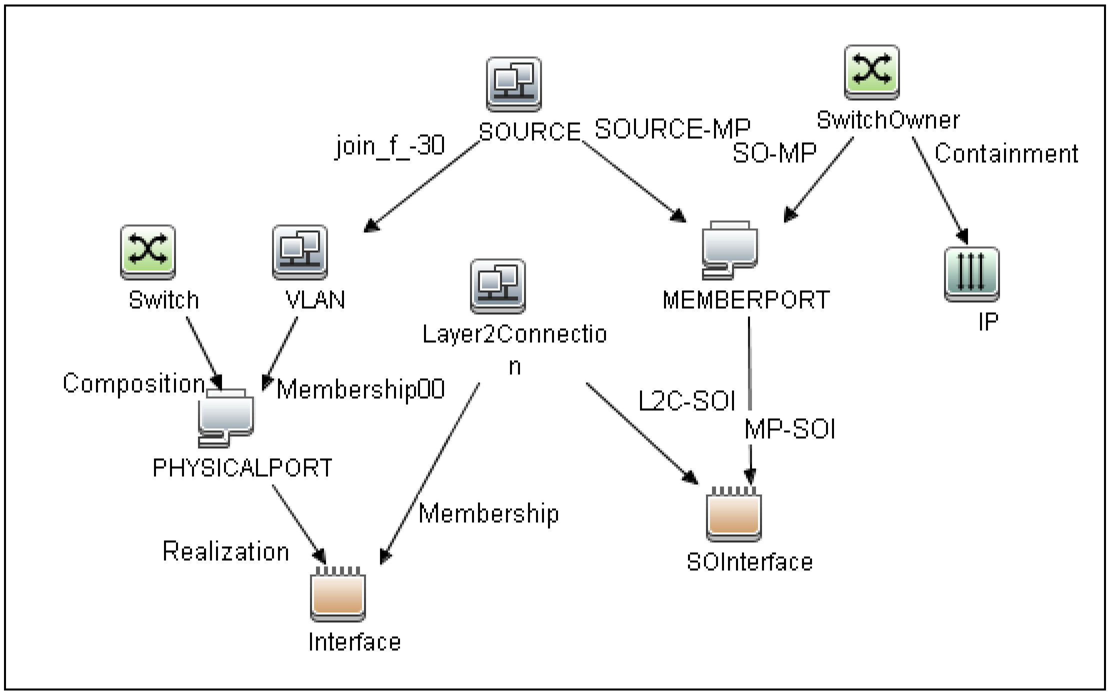
Merge VLANs Adapter

This section includes details about the adapter.

Input CIT

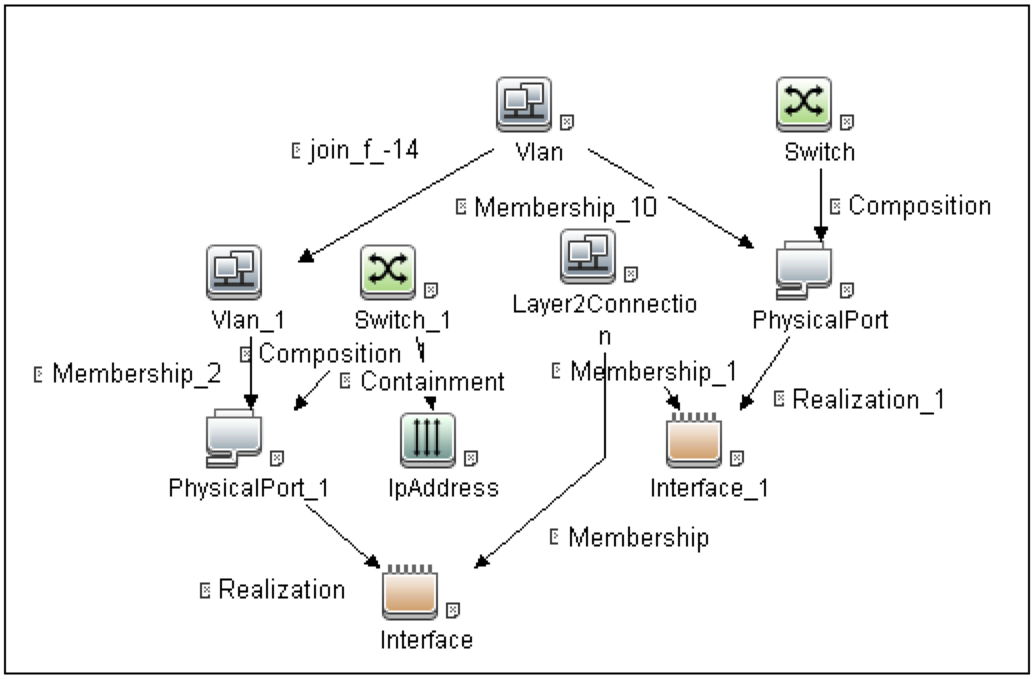
Vlan

Input Query



Node Name	Condition
IP	NOT IP Probe Name Is null

Trigger TQL Query



Node Name	Condition
IpAddress	NOT IP Probe Name Is null

Triggered CI Data

Name	Value
memberId	${MEMBERPORT.root_id}
portId	${PHYSICALPORT.root_id}
vlanId	${SOURCE.vlan_id}

Used Scripts

merge_vlans_by_ports.py

Discovered CITs

- Membership

- PhysicalPort
- Vlan

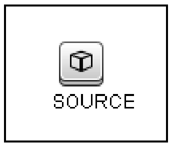
Process Layer2 Collected Files Adapter

This section includes details about the adapter.

Input CIT

Discovery Probe Manager

Input TQL Query



Used Script

processL2Files.py

Discovered CITs

- Composition
- Interface
- Layer2Connection
- Membership
- Node

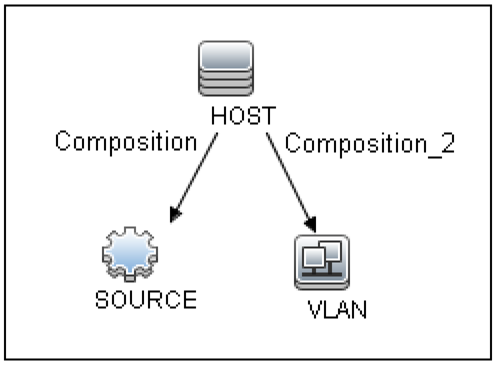
VMS Catalyst by SNMP Adapter

This section includes details about the adapter.

Input CIT

SNMP

Input TQL Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
hostId	${SOURCE.application_ip_domain}
ip_address	${HOST.root_id}
snmpCommunityPostfix	${VLAN.vlan_id}

Used Scripts

SNMP_Net_Dis_VMS_catalyst.py

Discovered CITs

- Bridge

- Composition
- Containment
- Dependency
- Membership
- PhysicalPort
- Vlan

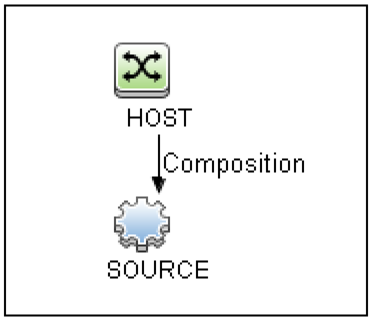
Catalyst Vlans by SNMP Adapter

This section includes details about the adapter.

Input CIT

SNMP

Input TQL Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}

Used Scripts

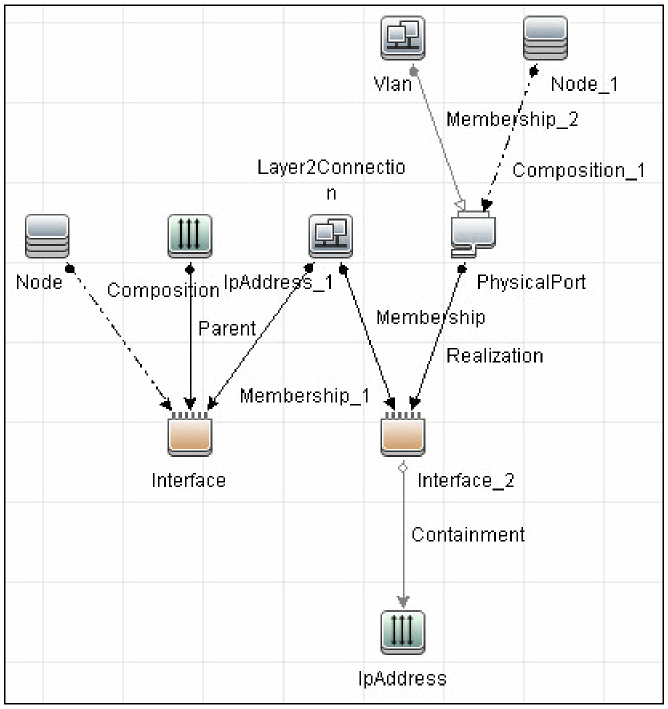
SNMP_Net_Dis_Catalyst_Vlans.py

Discovered CITs

- Bcast Domain
- Composition
- ELAN
- ELAN-VLAN Map
- Membership
- PhysicalPort
- Vlan

Relationships

- A Layer2 switch can be connected to its ports directly or through a VLAN.
- The Bridge CIT represents the basic MAC address (Network Interface Card) on which the ports are located.
- Each port on the switch can be connected to a host or interface object (the end user machines) by a Layer2 CI, or to a port-switch by a Backbone link.



Troubleshooting and Limitations

This section describes troubleshooting and limitations for Layer2 discovery.

- If the results of the discovery return empty, verify that you have access to the discovered SNMP agent (or to the SNMP agent using the special community authentication) and that all the requested MIB tables are responding to SNMP requests from the Data Flow Probe machine. For details on the MIB tables, refer to the appropriate script.

- In cases where the reported bridge MAC address is `000000000000`, "", or `null`, the adapter does not report results.

- If the retrieved basic bridge MAC (retrieved from the `1.3.6.1.2.1.17.1.1` table) is not the same as the given `bridgeId` in the destination data, the adapter returns zero results.
  In the case of SNMP_Dis_L2_Bridge, `bridgeId` is set by bridge_basemacaddr.
  In the case of SNMP_Dis_L2_VLAN, `bridgeId` is set by `vlan_bridgemac`.

Chapter 73: No-Credentials Discovery

This chapter includes:

Overview

Nmap is a utility for network exploration that uses raw IP packets to determine which hosts are available on the network, which services those hosts are offering, which operating systems they are running on, and so on.

Nmap also calculates to what extent the operating system result is accurate - for example, 80% accuracy.

DFM uses the nmap utility in the following jobs:

- **Host Fingerprint using nmap**

  This job reports the Nmap accuracy value on the `host_osaccuracy` attribute on the Host CI.

- **Range IPs by NMAP**

  This job reports on live hosts.

How to Set Up the Data Flow Probe Machine

Perform the following procedure on every Data Flow Probe machine that is to run either or both of the **Host Fingerprint using nmap** or **Range IPs by nmap** jobs.

1. Run **nmap-4.76-setup.exe** from **C:\hp\UCMDB\DataFlowProbe\tools**.
2. Accept the terms of the license and click **I agree**. The **Choose Components** dialog box opens.
3. Select **Nmap Core Files**, **Register Nmap Path**, and **WinPcap 4.2.1**.
4. Click **Next**.

   The **Choose Install Location** dialog box opens.
5. Accept the default location or enter another location. Click **Install**.

   Nmap is installed. The WinPcap installation dialog box opens immediately after the Nmap installation is complete.
6. Accept the terms of the license and click **Next**. The **Choose Install Location** dialog box opens.
7. Accept the default location or enter another location. Click **Install**.

   The Finished dialog box opens.
8. Click **Finish**. The WinPcap Options dialog box opens.
9. Clear the check boxes and click **Next**.
10. Click **Finish**.

    The following software is added to the Data Flow Probe machine:

    - Nmap 6.2.5
    - winpcap-nmap 4.1.2
    - Microsoft Visual C++ Redistributable - x86 2010

   To verify, access the **Add/Remove Programs** window.

How to Discover Host Fingerprint with nmap

This task describes how to use the **Host Fingerprint using nmap** job to discover hosts, operating systems, network interfaces, applications, and running services.

This task includes the following steps:

1. **Prerequisites- Set up protocol credentials**

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

2. **Prerequisites - Set up Data Flow Probe machine**

   See "How to Set Up the Data Flow Probe Machine" on the previous page.

3. **Run the discovery**

   This job is triggered on any discovered IP address.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

How to Discover Range IPs by nmap

This task includes the following steps:

1. **Prerequisites - Set up Data Flow Probe machine**

   See "How to Set Up the Data Flow Probe Machine" on the previous page.

2. **Run the discovery**

   Run the **Range IPs by nmap** job.

   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

Host Fingerprint using nmap Job

Adapter

This job uses the **OS_Fingerprint** adapter.

Parameters

To view the parameters, go to **Universal Discovery > Discovery Modules/Jobs > Network Infrastructure > No-Credentials Discovery > Host Fingerprint using nmap > Properties tab > Parameters pane**.

For details on overriding parameters, see "Parameters Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Parameter	Description
Create_Application_CI	**True.** Creates an application CI based on the port fingerprint information.
Perform_Port_Fingerprints	**True.** Tries to discover opened ports.
discover_os_name	**True.** Discovers host OS, which may have some inaccuracy.
nmap_host_timeout	The length of time nmap is allowed to spend scanning a single host (in seconds).
nmap_location	Full path to nmap executable file. Example: **C:\Program Files (x86)\Nmap\nmap.exe** **Note:** If empty, the job looks in the system path.
scan_known_ports_only	Scans for ports listed in the **portNumberToPortName.xml** file. **Default:** False

Parameter	Description
scan_these_ports_only	Limits the range of ports to be scanned. For example: `T:1-10,42,U:1-30` (discover TCP ports 1 to 10 and 42 and UDP ports 1-30). If this parameter is left empty, the Nmap default is used.

Discovered CITs

To view discovered CITs, select a specific adapter in the Resources pane.

For details, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Range IPs by nmap Job

Adapter

This job uses the **IpRange_by_nmap** adapter.

Parameters

To view the parameters, go to **Universal Discovery > Discovery Modules/Jobs > Network Infrastructure > Basic > Range IPs by nmap > Properties tab > Parameters pane**.

For details on overriding parameters, see "Parameters Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Name	Description
excludePatternsList	A list of wildcard patterns, separated by semicolons. IP addresses matching any of the patterns are skipped. The pattern may include numbers, dots, or the wildcards ***** (matches zero or more characters) or **?** (matches exactly one character).
nmap_location	Full path to the **nmap** executable file. Example: **C:\Program Files (x86)\Nmap\nmap.exe** **Note:** If empty, the job looks in the system path.
range	A range of IPs to ping, separated by a semicolon. For example: 1.2.3.0-1.2.3.10;1.2.3.50-1.2.3.60

Discovered CITs

IpAddress

Discovery Flow

The discovery is performed for each range specified in the probe, as follows:

1. Filter IPs in range, applying the patterns specified in the **excludePatternsList** parameter.

2. Perform ping scan on filtered IPs.

> **Note:** Nmap performs a ping scan of filtered IPs, 10 at a time. So, if 100 IPs are passed to check, the nmap command executes 10 times. This is because of command line size limitations, especially in Windows.

3. Send live IPs to UCMDB before processing the next range.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for No-Credentials discovery.

Error Message	**Reason**	**Solution**
Can't parse XML document with Nmap results. Skipped.	nmap.exe failed before it could create a valid XML file.	• Try to restart the Nmap job. • Try to reduce the number of threads for the Nmap job.
Error nmap result file is missing	nmap.exe failed before it could create an XML file.	• Try to restart the Nmap job. • Try to reduce the number of threads for the Nmap job.
The system cannot execute the specified program (in the communication log file)	The Windows system cannot launch the Nmap application.	Verify that: • The correct Nmap version has been downloaded and installed. • WinPcap has been installed. For details on these installations, see "Prerequisites- Set up protocol credentials" on page 1036. If you have installed Nmap and WinPcap, and the error message still appears in the communication log, install **vcredist_x86.exe** from **C:\hp\UCMDB\DataFlowProbe\runtime \probeManager\discoveryResources**.
Nmap is not installed on Probe machine	Nmap is not installed on the Probe machine.	Try to launch Nmap from the command line. Make sure that Nmap is installed. For details on the installation, see "Prerequisites- Set up protocol credentials" on page 1036.

Chapter 74: Active and Passive Network Connections Discovery

This chapter includes:

Overview

All jobs in these modules run queries against the Data Flow Probe's PostgreSQL database to retrieve network connectivity information inserted by the **Host Resources and Applications** and/or **TCP By Shell/SNMP** and/or **Collect Network Data by Netflow** jobs.

For details on Host Resource jobs, see "Host Resources and Applications Discovery" on page 698.

The Data Flow Probe includes a built-in PostgreSQL database so there is no need to install a separate PostgreSQL instance for NetFlow. Instead, data is saved to a dedicated scheme (called `netflow` for historical reasons).

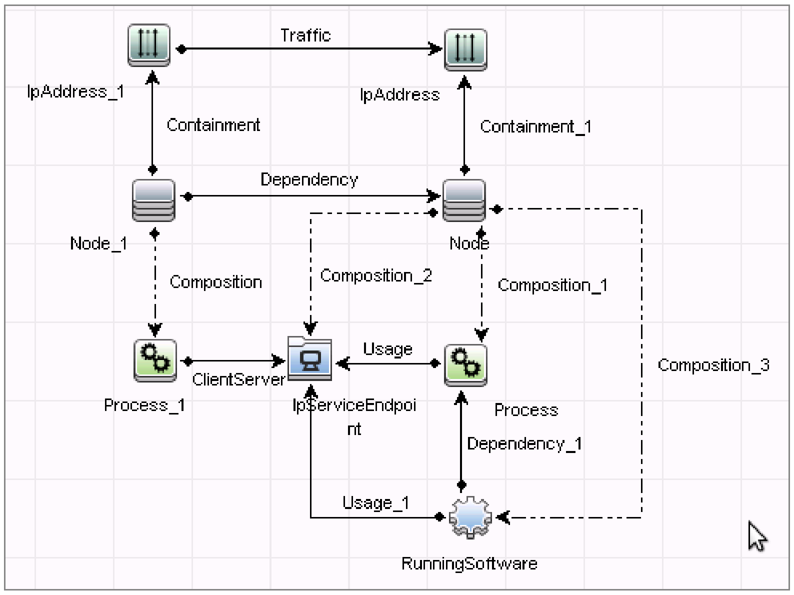
Supported Versions

This discovery supports NetFlow versions 5 and 9.

Note: By default, support for NetFlow versions is enabled. To disable support for any version, set the **flow.collector.V{netflow_version}.enabled** property in the **NetFlow.properties** file to false.

Topology

Network Connection Passive Discovery



How to Discover Processes

This task describes how to discover processes.

This task includes the following steps:

1. **Prerequisite - Set up protocol credentials**

   To discover network connections, define the following protocols:

   - SNMP protocol
   - NTCMD protocol
   - SSH protocol
   - Telnet protocol
   - WMI protocol

   For credential information, see "Supported Protocols" in the *HP Universal CMDB Discovery and Integration Content Guide - Supported Content* document.

   Note: None of these protocols is mandatory, but WMI alone does not retrieve network data.

2. **Run the discovery**

   Run the following jobs in the following order:

   - Run the **TCP Data by Shell** or **TCP Data by SNMP** job to populate the Probe's PostgreSQL database with TCP information gathered from the remote machine. For details, see "TCP Traffic Jobs" on the next page.
   - Run the **Network Connectivity Data Analyzer** job. For job details, see "Network Connectivity Data Analyzer Job" on page 1047.

TCP Traffic Jobs

The **TCP Data by Shell** and **TCP Data by SNMP** jobs enable you to collect information about TCP traffic. These jobs do not send CIs to the CMDB but run queries against existing data in the Data Flow Probe's database.

These jobs are enhanced with the following parameters that enable you to capture TCP data and to configure the time delay between captures:

Parameter	Description
CaptureProcessInformation	**true**: process information is captured and stored in the Data Flow Probe's database. No CIs are reported. Processes are captured with the same method as that used by the Host Resources and Applications job. For details on Host Resource jobs, see "Host Resources and Applications Discovery" in the *HP UCMDB Universal Discovery Content Guide - Discovery Modules* document.
DelayBetweenTCPSnapshots	The number of seconds between TCP snapshot captures. The default is 5 seconds. It can be useful to take several TCP snapshots during a single job invocation, to retrieve more detailed data. For example, when running the **netstat -noa** command on a remote Windows system to gather TCP information, this parameter can capture process information at 5-second intervals during the command run.
NumberOfTCPSnapshots	The number of TCP snapshots to take.
lsofPath	The path to the `lsof` command that enables process communication discovery on UNIX machines. The default value is **/usr/local/bin/lsof,lsof,/bin/lsof**.
useLSOF	**true:** discovery tries to use **lsof** utility to discover port-to-process mappings on UNIX machines. **Default:** True
useNetstatOnly	Specifies whether or not to run additional commands (lsof and pfiles) or to use the netstat command only. **Default:** False

Network Connectivity Data Analyzer Job

This job allows users to capture TCP communication information from the IT Server infrastructure and model them inside the UCMDB. It can be configured to report customized topology. For details, see "TcpDiscoveryDescriptor.xml File" on page 1050.

Adapter

This job uses the Network_Connectivity_Data_Analyzer adapter.

- Adapter Parameters

Parameter (A-Z)	Description
acceptedServices	Lists the services to be reported (ssh, oracle, mysql, and so on). When the value is set to **known_services**, a server running software is reported only if the service it represents is configured as **discover="1"** in the **portNumberToPortName.xml** file. When the value is set to '*****', all found services are reported.
discoveryDescriptorFile	The full path to a job configuration file used to define the analysis and reporting approach per IP range scope.
includeOutscopeClients	**True.** Enables reporting of outscope clients. **False.** Disables reporting of outscope clients.
includeOutscopeServers	**True.** Enables reporting of outscope servers. **False.** Disables reporting of outscope servers.
reportIpTrafficLink	**True.** Enables reporting of traffic link. **False.** Disables reporting of traffic link.
reportNodeDependencyLink	**True.** Enables reporting of dependency link. **False.** Disables reporting of dependency link.
reportServerRunningSoftware	**True.** Enables reporting of server running software. **False.** Disables reporting of server running software.

Discovered CITs

- Client-Server. DFM determines which machine is the server and which is the client:
  - If one end is discovered as a listening port, then this end is presumed to be a server.
  - If one end fits the minimal condition of **StatisticBasedApproach** (see server detection

approaches section) it is presumed to be a server.

   - If both ends have just one connection to a port, DFM identifies whether the end is a server by checking the ports and the **portNumberToPortName.xml** file (**Adapter Management > Resources pane > Packages > DDMInfra > Configuration Files**).

- Composition
- Containment
- Dependency. Link is set between discovered client and server.
- IpAddress
- IpServiceEndpoint
- Node
- Process
- Traffic. Link is set between IP addresses.
- Usage

TcpDiscoveryDescriptor.xml File

The **TcpDiscoveryDescriptor.xml** file defines rules for analysis and reporting per IP range scope.

This section includes:

Server Detection Approaches

The **serverDetectionApproach** tag contains a list of approaches used to resolve client server relation.

ListenPortsBasedApproach	Resolves a relation based on the **LISTEN** or **ESTABLISHED** connection state. It is necessary to run process-to-process discovery to be able to use that approach. If the port is opened for listening the host is resolved as server, so the second member of a connection is resolved as client automatically; and vice versa.
KnownPortsBasedApproach	Resolves a relation based on known a server port list defined in the **portNumberToPortName.xml** file.
StatisticBasedApproach	Resolves a relation based on a minimal condition. If the condition value is zero it is not taken in to account. Valid conditions are: • **minClients.** Minimum connections count to indicate host as a server. • **minPackets.** Minimum total packets count sent and received by a host to indicate it as a server. • **minOctets**. Minimum total octets count sent and received by a host to indicate it as a server.

Note: An approach can be deactivated if its active attribute is set to **false** or the tag responsible for the approach is commented out or removed.

Filtering

The **Filtering** section defines filter rules applied to discovered clients and servers.

Note: A host is filtered if at least one of the filters is applied to it.

The filter types are:

- "Range Filter" below
- "Service Filter" below

Range Filter

The Range filter performs filtering on a per-IP-range basis.

Example:

range filter definition

```
<ranges>
    <include>
        <range>probe_ranges</range>
    </include>
    <exclude>
        <range>outscope_clients</range>
    </exclude>
</ranges>
```

Ranges that must be included in the final reporting topology should be defined in the **<include>** tag. Ranges that must be excluded should be defined in **<exclude>** tag. The following keywords should be used to define specific ranges:

Keyword	Description
probe_ranges	Includes all ranges defined using the Protocol Manager.
outscope_clients	Includes all client IPs that are out of Probe range scope.
outscope_servers	Includes all server IPs that are out of Probe range scope.
ddm_related_ connections	Includes the Probe IP. Allows user to filter DFM-related connections initiated during the discovery process.

Service Filter

The Service filter performs filtering of discovered servers according to the specified list of services.

Mapping between service name and relevant port is done according to definitions in the **portNumberToPortName.xml** file.

Example:

range filter definition

```
<services>
    <include>
        <service name="*" />
    </include>
    <exclude>
        <service name="ssh" />
    </exclude>
</services>
```

Services that must be included in final reporting topology are defined in **<include>** tag. Services that must be excluded are defined in **<exclude>** tag. When the **service name** value is "*" (asterisk), all servers found.

Note: A service can be deactivated if its active attribute is set to **false** or the tag responsible for the service is commented out or removed.

Reporting

The **Reporting** section is responsible for defining filter rules and lists of active reporters. The **configuration** tag defines default filtering rules for all the reporters. A reporter can override a filtering rule by defining the **<filtering>** tag in its body. Each reporter is responsible for the topology being reported.

Note: A reporter can be deactivated if its active attribute is set to **false** or the tag responsible for the reporter is commented out or removed.

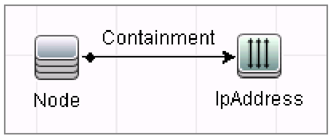
The following reporters are available:

- **Default**. For details, see "Default Reporter" on the next page.
- **clientProcess**. For details, see "Client Process Reporter" on page 1054.
- **clientServerLink.** For details, see "Client Server Link Reporter" on page 1055.
- **ipTrafficLink.** For details, see "IP Traffic Link Reporter" on page 1056.
- **nodeDependencyLink.** For details, see "Node Dependency Link Reporter" on page 1057.

- **serverProcess.** For details, see "Server Process Reporter" on page 1058.
- **serverRunningSoftware.** For details, see "Server Running Software Reporter" on page 1059.

- ## Default Reporter

If no reporters are activated, the job returns the **IP** and **Node** CIs linked by the **containment** relationship only.



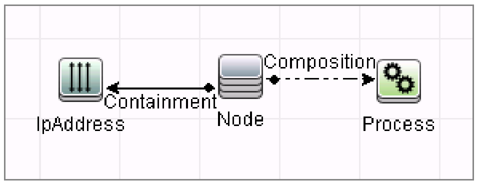
- ## Client Process Reporter

This reporter reports client processes.

reporter definition

```
<reporting>
    <reporter name="clientProcess" active="true"/>
</reporting>
```

Topology



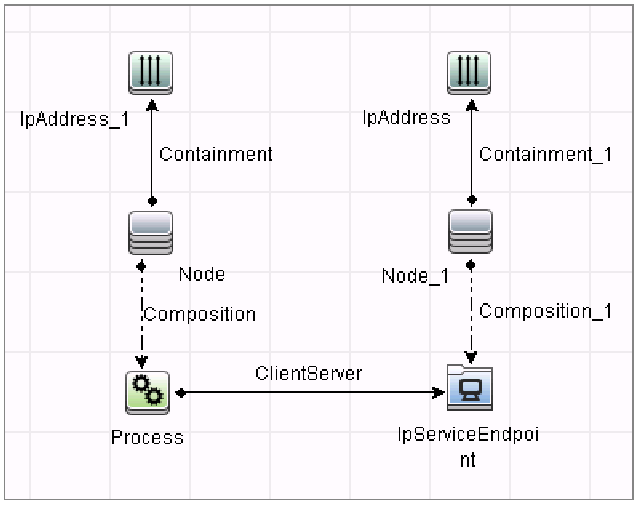
- **Client Server Link Reporter**

This reporter reports the client process communication endpoint and the client-server link between them (even if clientProcess active="false").

reporter definition

```
<reporting>
    <reporter name="clientServerLink" active="true"/>
</reporting>
```

Topology



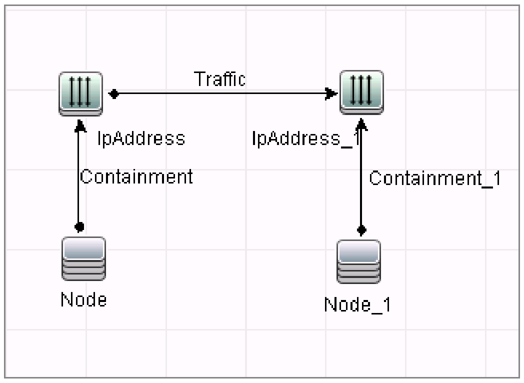
- ## IP Traffic Link Reporter

This reporter the traffic link between IPs. The **reportTrafficDetails** attribute indicates whether the job should report the **octetCount**, **packetCount** and **portset** attributes of the link.

reporter definition

```
<reporting>
    <reporter name="ipTrafficLink" active="true" reportTrafficDetails="true"/>
</reporting>
```

Topology



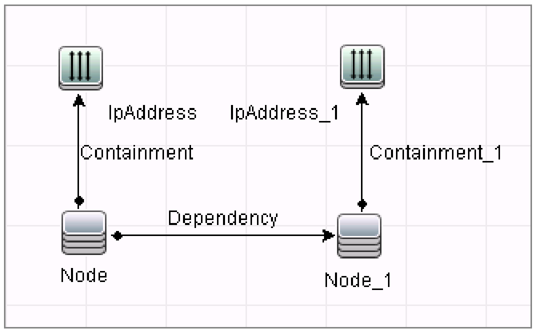
- **Node Dependency Link Reporter**

This reporter reports the dependency link between discovered nodes.

reporter definition

```
<reporting>
    <reporter name="nodeDependencyLink" active="true"/>
</reporting>
```

Topology



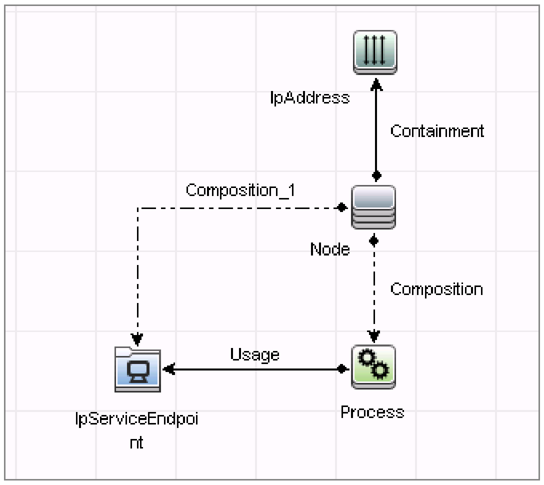
- **Server Process Reporter**

This reporter reports the server process. The **linkWithCommunicationEndpoint** attribute indicates whether the reporter should link the process with the discovered communication endpoint (with 'usage' link).

reporter definition

```
<reporting>
    <reporter name="serverProcess" active="true" linkWithCommunicationEndpoint="true"/>
</reporting>
```

Topology



- ## Server Running Software Reporter

This reporter reports server running software linked with communication endpoint (with 'usage' link) and server process. The **linkWithProcess** attribute indicates whether the reporter should link the discovered running software with the server process (with '**dependency**' link).

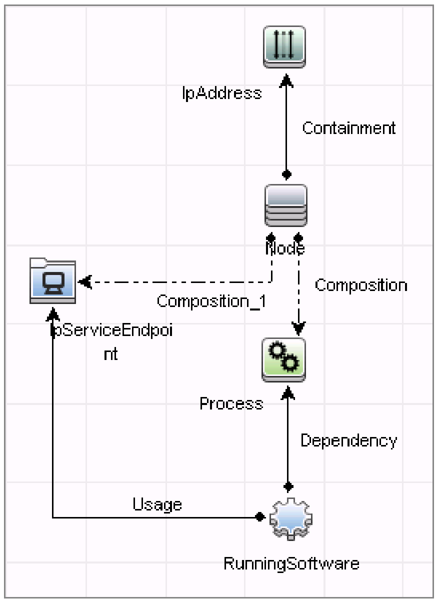
The reporting of server running software is dependent on the **acceptedServices** parameter:

- If set to **known_services**, a server running software is reported only if the service it represents is configured as **discover="1"** in the **portNumberToPortName.xml** file.
- If set to '*', all found services are reported.
- If it contains any TCP ports or service names, the job reports only the running software that works with these ports.

reporter definition

```
<reporting>
    <reporter name="serverRunningSoftware" active="true" linkWithProcess="true"/>
</reporting>
```

Topology



Part 16: Tools and Samples > Discovery Tools

Chapter 75: File Monitor by Shell Job

This section includes details about the File Monitor by Shell job.

Trigger Query

None

Job Parameters

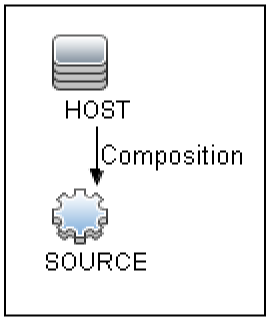
Parameters are not overridden by default and use values from the adapter.

Adapter Information

This job uses the **FILE_Mon** adapter.

Input CIT

Shell



Triggered CI Data

Name	Value
codepage	${SOURCE.codepage:NA}
connected_os_credentials_id	${SOURCE.connected_os_credentials_id:NA}
credentialsId	${SOURCE.credentials_id}

Name	Value
hostId	${HOST.root_id}
ip_address	${SOURCE.application_ip}
language	${SOURCE.language:NA}
Protocol	${SOURCE.root_class}

Used Scripts

- file_ver_lib.py
- file_mon_utils.py
- file_mon.py

Discovered CITs

- Composition
- ConfigurationDocument

Adapter Parameters

Name	Default Value	Description
binary_file_extensions	exe, bin, dat	This parameter defines what type of files are binary so that content is not read from those files.
discoverUnixHiddenFiles	false	This flag determines whether to discover hidden files on Unix-like systems. If **true** then all hidden files are discovered. This parameter is not dependent on the values assigned to the extensions parameter.
extensions	cfg, conf, txt, xml, exe, bin, dat	This parameter defines the files extensions that the discovery looks for.
folders	C:\temp,D:\temp	A list of folders from which to gather files.

Name	Default Value	Description
recursively	false	This flag determines whether to scan folders recursively.

Global Configuration File

globalSettings.xml

Troubleshooting and Limitations

This section describes troubleshooting and limitations for file discovery, when running the **File Monitor by Shell** job.

- The **File Monitor by Shell** does not trigger automatically. This is because there is no trigger TQL query for this job: an automatic trigger on all destinations may cause an out-of-memory error on the Data Flow Probe. To solve this problem, add the triggered CI manually.
- When running the **File Monitor by Shell** job, discovering files of more than 2Mb may cause an out-of-memory error.

Chapter 76: HP Serviceguard and Oracle RAC Discovery

This chapter includes:

Overview

This job is a part of the support for HP Serviceguard and Oracle RAC. The introduced mechanism allows reporting of an indirect link between an Oracle database instance and the HP Serviceguard package through FS resources.

Supported Versions

This job supports HP-UX 10 and HP-UX 11 with Oracle RAC 10i.

How to Run the Link DB DataFiles and Clustered FS Job

1. **Prerequisites**

   The job does not require any credentials, because it is simply a complex enrichment. Therefore, the only prerequisite is that the particular topology should be present in UCMDB to make the job trigger.

2. **Run the discovery**

   Run the following jobs:

   a. Run the **Range IPs by ICMP** job.

   b. Run the **Host Connection by Shell** job.

   c. Run the **Host Resources by Shell** and **Host Applications by Shell** jobs.

   d. Run the **Service Guard Cluster Topology** job.

   e. Run the **Oracle Topology by SQL** job.

   f. Run the **Link DB DataFiles And Clustered FS** job.

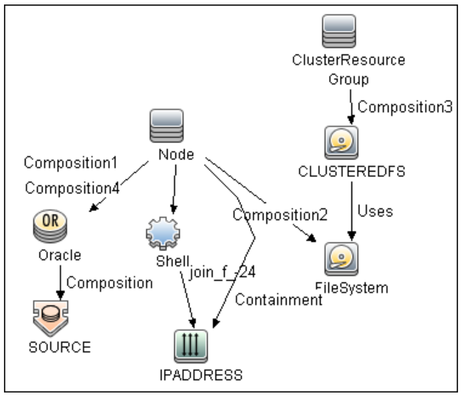
      For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide.*

Adapter

Input CIT

DB Data File

Input Query



Triggered CI Data

Name	Value
dbFileId	${SOURCE.root_id}
dbFilePath	${SOURCE.name}
fsId	${CLUSTEREDFS.root_id}
mountPoints	${CLUSTEREDFS.mount_point}

Used Script

- linkDbDatafileAndFs.py

Discovered CITs

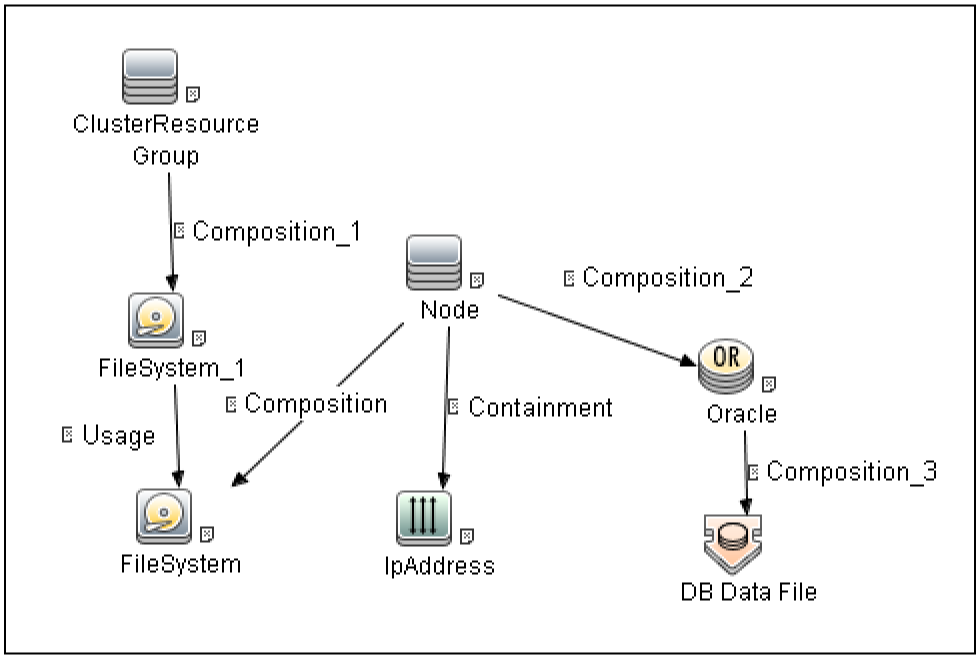
- DB Data File
- FileSystem

- Node
- Usage

Link DB DataFiles and Clustered FS Job

This section includes details about the job.

Trigger Query



Discovery Flow

The approach for linking DB Data File and File System is as follows:

1. The **Service Guard Cluster Topology by TTY** job reports File System Objects that are mount points of the Serviceguard package. So, these are File System package resources.

2. The **Oracle Topology by SQL** job reports Oracle DB and DB Data Files.

3. Where there is a topology in which ClusteredResource Groups has FS resources, and on at least one node of this cluster there is a running Oracle database with discovered DB Data Files, the **Link DB DataFiles and Clustered FS** job looks at all mount point and DB Data Files. The job finds valid relationships between them, if any, and reports each as a new link.

Chapter 77: Merge Clustered Software

This chapter includes:

Overview

This document describes the usage and functionality of the **Merge_Clustered_software** discovery package. The package makes it possible to merge CIs which show the presence of a particular RunningSoftware on a cluster Node with a Clustered Service.

Supported Software

This discovery package supports the discovery of:

- HP Serviceguard Cluster with:
  - Oracle Database;
  - Oracle TNS Listener; and
  - Oracle iAS
- Microsoft Cluster Server (MSCS) with:
  - Microsoft SQL Server

How to Merge Clustered Software

In the Data Control Panel, activate the discovery job.

For details on running jobs, see "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

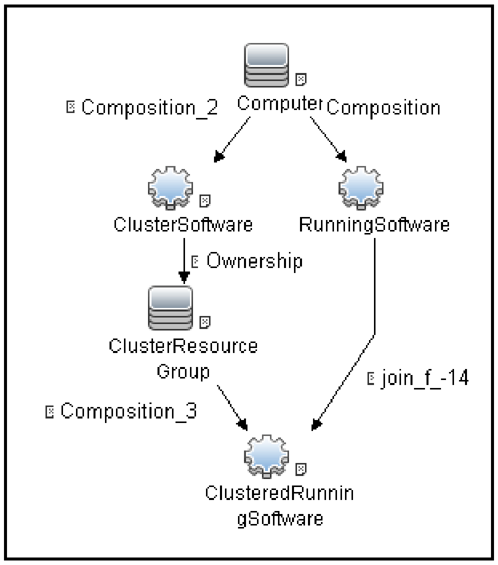
Note: To widen the scope of the discovery, the user should update the Trigger TQL Query and the Input TQL Query by adding the appropriate CIT names to the parameters. No additional changes are required.

Merge Clustered Software Job

This section includes details about the job.

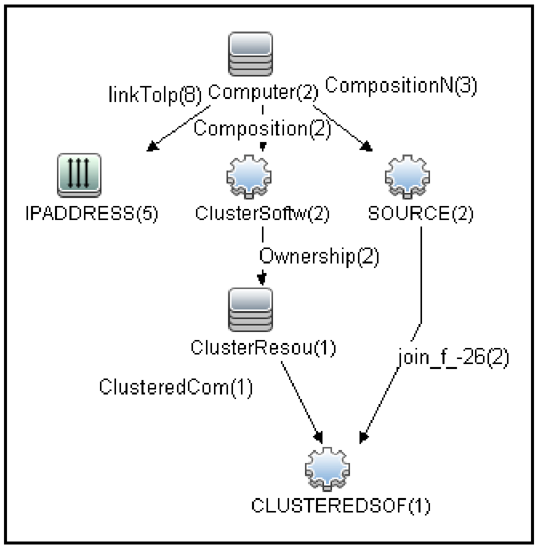
Trigger TQL Query

The following graphic shows the Trigger TQL Query for merging clustered software.



Input TQL Query

The following graphic shows an Input TQL Query for merging clustered software.



Triggered CI Data

- className
- clusteredContainer
- clusteredUcmdbIds
- discProdName
- localSoftwareId
- productName
- softwareName

Discovered CITs

- Composition
- Node
- RunningSoftware

Used Script

- mergeClusteredSoftware.py

reated/Changed Entities

Entity Name	Type	Description
mergeClusteredSoftware.py	Script	Discovery Script
Merge_Clustered_Software.xml	Pattern	Discovery Pattern
Merge Clustered Software.xml	Job	Discovery Job
mergeDiscClusteredSoft.xml	TQL Query	Trigger TQL Query

Chapter 78: TCP/UDP Ports Discovery by Nmap

This chapter includes:

Overview

This discovery allows you to discover, on a particular host, the open TCP or UDP ports of the known server ports.

Supported Versions

This discovery supports Nmap version 6.25 and later.

How to Discover TCP/UDP Ports by Nmap

1. **Prerequisites - Set up the Data Flow Probe machine**

   Perform the following procedure on every Data Flow Probe machine that is to run the TCP Ports job.

   a. Run **nmap-6.25-setup.exe** from **C:\hp\UCMDB\DataFlowProbe\tools**.

   b. Accept the terms of the license and click **I agree**. The **Choose Components** dialog box opens.

   c. Select **Nmap Core Files**, **Register Nmap Path**, and **WinPcap 4.1.2**.

   d. Click **Next**.

      The **Choose Install Location** dialog box opens.

   e. Accept the default location or enter another location. Click **Install**.

      Nmap is installed. The WinPcap installation dialog box opens immediately after the Nmap installation is complete.

   f. Accept the terms of the license and click **Next**. The **Choose Install Location** dialog box opens.

   g. Accept the default location or enter another location. Click **Install**.

      The Finished dialog box opens.

   h. Click **Finish**. The WinPcap Options dialog box opens.

   i. Clear the check boxes and click **Next**.

j. Click **Finish**.

   The following software is added to the Data Flow Probe machine:

   - Nmap 6.25
   - winpcap-nmap 4.1.2
   - Microsoft Visual C++ Redistributable - x86 2010

   To verify, access the **Add/Remove Programs** window.

2. **Run the discovery**

   Run the following jobs:

   a. **Range IPs by ICMP** or **Range IPs by nmap** to discover which of the machines in the IP range are up.

   b. **TCP Ports** job.

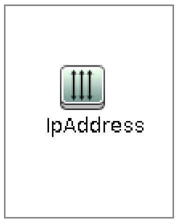
   For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide.*

TCP Ports Job

Adapter

This job uses the **TCP Ports Discovery** adapter.

Trigger Query



Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

Job Parameters

Name	Default Value	Description
checkIfIpIsReachable	true	This flag indicates whether the job should check if the discovered IP is reachable before the job starts to check availability of the host's ports.
checkOnlyKnownPorts	true	This flag indicates whether the job should discover only known ports. This flag does not cancel the **ports** or **UDPports** parameters. Setting this flag to **false** is applicable only with a real port range in the **ports** or **UDPports** parameter.
connectTimeOut	5000	The timeout (in milliseconds) when connecting to an IP and port.
nmapPath		The full path to the nmap executable file (for example: **C:\Program Files (x86)\Nmap\nmap.exe**).
pingTimeOut	2000	The ICMP ping timeout (in milliseconds).

Name	Default Value	Description
ports	**For JEE TCP Ports job:** <ul><li>weblogic, weblogicSSL, websphere_jmx, rmi</li></ul>**For Database TCP Ports job:** <ul><li>oracle, db2, sybase, sql, mysql</li></ul>**For SAP TCP Ports job:** <ul><li>sap, sap_jmx, sap_http, sap_https</li></ul>**For SAP TCP Ports job:** no default value	This parameter contains a list of TCP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known TCP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.
scanUDP	false	This flag indicates whether or not to scan UDP ports. **Note:** UDP scanning is supported only if **useNMap=true** (see below).
UDPports		This parameter contains a list of UDP ports on which discovery is performed. This list can include ranges, separate port numbers, and known protocol names (such as http, ftp, etc.) and must be comma separated. If this list is empty or contains the value *****, discovery is performed only on all known UDP ports. If a port range is entered (such as 1000-1100), discovery is performed only on all known ports in that range if **checkOnlyKnownPorts=true**.

Name	Default Value	Description
useNMap	**For Database TCP Ports and JEE TCP Ports jobs:** false **For SAP TCP Ports and TCP Ports jobs:** true	This flag indicates whether or not to use nmap during port scanning. **Note:** If no path is specified for **nmapPath** (see above), the nmap from the system path is used.

Note: Only ports on which a port name has been assigned to it in the **ports** or **UDPports** parameters and which are marked as 'discoverable' (**isDiscovered=1**) in the **portNumberToPortName.xml** configuration file are discovered.

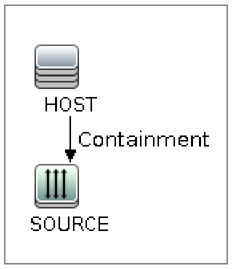
Adapter Information

This adapter discovers TCP ports.

Input CIT

IpAddress

Input Query



Triggered CI Data

Name	Value
ip_address	${SOURCE.name}
ip_domain	${SOURCE.routing_domain}

Used Scripts

- TcpPortScanner.py
- nmap.py

Global Configuration File

portNumberToPortName.xml

Discovered CITs

- Composition
- Containment
- IpAddress
- IpServiceEndpoint
- Node

Part 17: Tools and Samples > SSL Certificate Discovery

Chapter 79: SSL Certificate Discovery

This chapter includes:

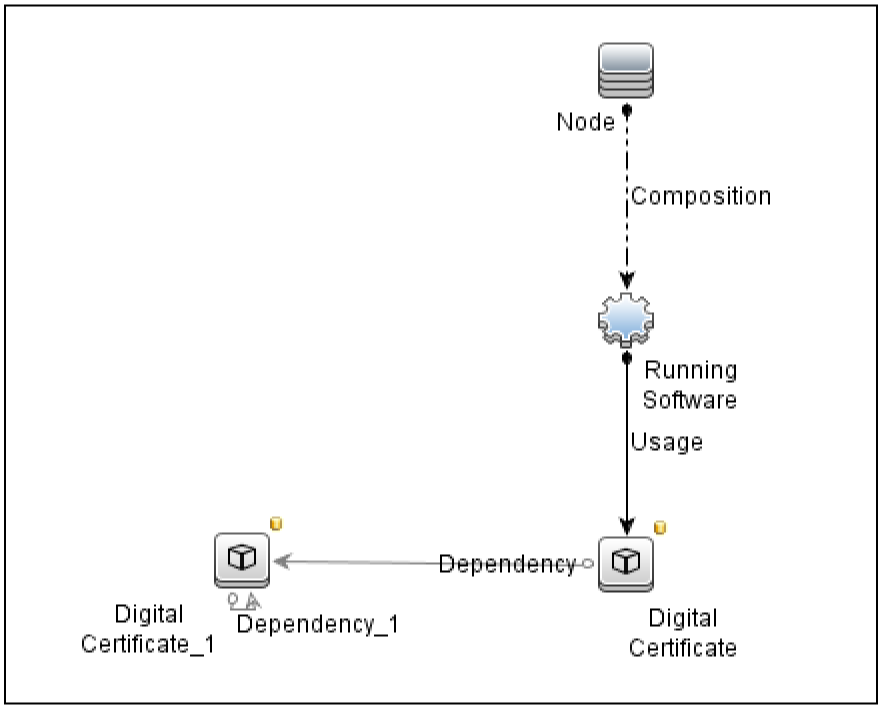
Overview

This discovery allows you to discover the SSL certificates used by any RunningSoftware CI which is reported as using the HTTPS protocol.

Supported Versions

This discovery supports the discovery of only those SSL certificates that comply with the X.509 standard for public key infrastructure and Privilege Management Infrastructure.

Topology



Discovery Mechanism

The discovery flow includes the following steps:

1. Connect to the host.
2. Perform SSL handshake.
3. Retrieve certificate information.

How to Discover SSL Certificates

1. **Prerequisites**

   For the discovery to succeed, there must be an **IpServiceEndpoint** CI that has a **IpServiceName** or **ServiceNames** attribute value of **https**.

2. **Run the discovery**

   - Run the following jobs in quick mode:

     Note: Quick mode discovers only known ports that are described in portNumberToPortName.xml. You can also run a more thorough discovery (see instructions below) that finds non-default ports that are discovered by application related jobs.

     i. **Range IPs by ICMP** job, to discover the target IPs.

     ii. Either **Web Server Detection using TCP Ports** job or **TCP Ports** job.

     iii. **SSL Certificate Discovery by HTTPS** job.

   - **Optional:** For a more thorough discovery, run the following jobs:

     i. **Host Connection by Shell** job, to discover the target host and shell connectivity to it.

     ii. **Host Applications by Shell** job, to discover applications of the target host.

     iii. **Optional:** Run any of the following jobs (according to the application you want to discover).

       - **Apache Tomcat by Shell**
       - **JEE Glassfish by Shell**
       - **JEE JBoss by Shell**
       - **JEE Weblogic by Shell**

- **JEE WebSphere by Shell**
- **Web Server by Shell**
- **IIS Applications by NTCMD or UDA**

iv. Run the **SSL Certificate Discovery by HTTPS** job.

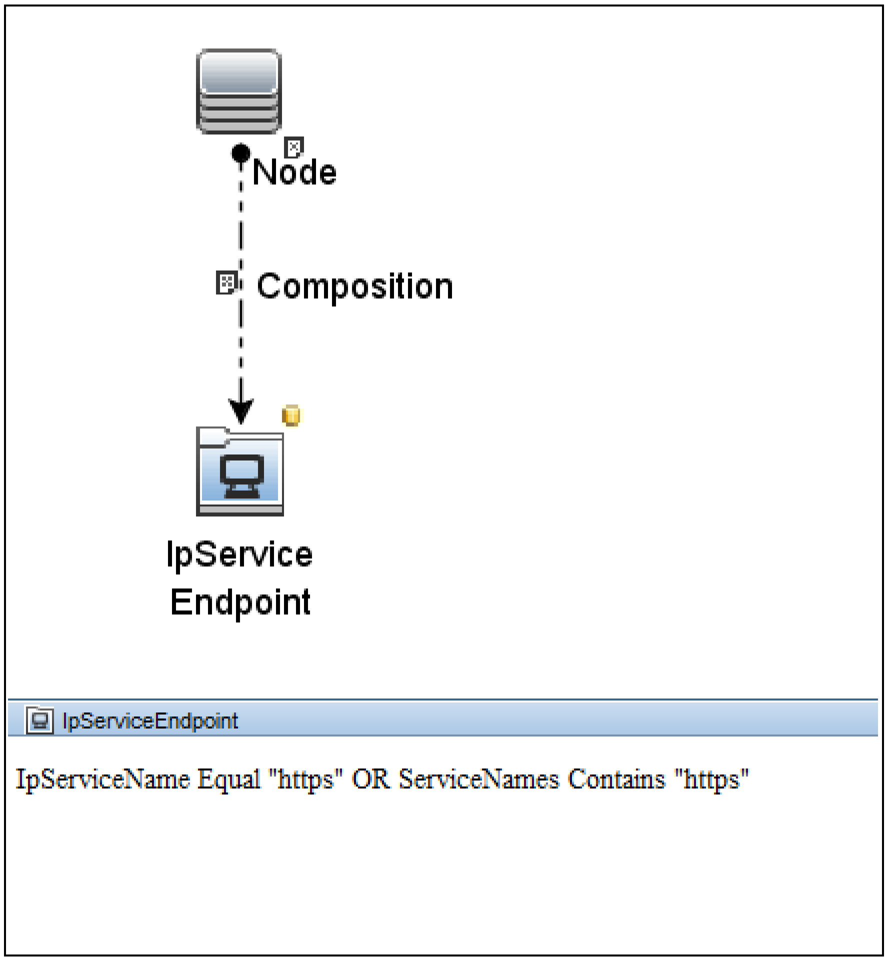
For details on running jobs, refer to "Module/Job-Based Discovery" in the *HP Universal CMDB Data Flow Management Guide*.

SSL Certificate Discovery by HTTPS Job

Adapter

This job uses the **SSL Certificate Discovery** adapter.

Trigger TQL



IpServiceName Equal "https" OR ServiceNames Contains "https"

Parameters

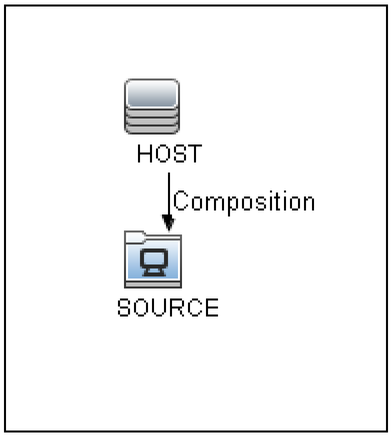
None

SSL Certificate Discovery Adapter

Input CIT

IpServiceEndpoint

Input TQL Query



Triggered CI Data

Name	Value
host_id	${HOST.root_id}
https_port	${SOURCE.network_port_number}
ip	${SOURCE.bound_to_ip_address}

Used Scripts

- ssl_cert.py
- ssl_cert_discoverer.py
- ssl_cert_discovery_by_https.py
- distinguished_name.py

Discovered CITs

- Dependency
- Digital Certificate
- IpServiceEndpoint
- RunningSoftware
- Usage

Parameters

None

Additional Information

The following jobs discover the name of a port. If this port is an https port, it can be used to trigger SSL Certificate Discovery:

- IIS Discovery by Shell
- Apache TomCat Discovery by Shel
- Apache Discovery by Shell
- GlassFish Discovery
- JBoss discovery
- Weblogic discovery
- WebSphere discovery
- SAP Discovery

Troubleshooting and Limitations

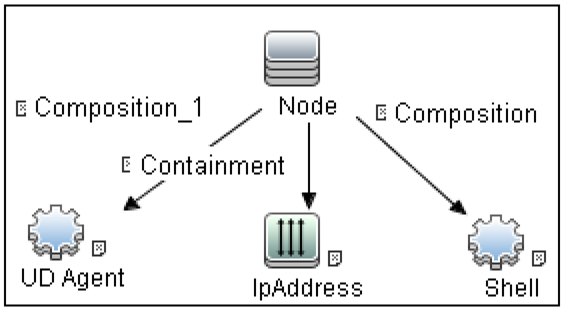
This job cannot retrieve the Certificate Authority for a certificate that is automatically trusted by the browser. In such cases, some root certificates do not appear in UCMDB.

Part 18: Tools and Samples > UD Agent Management

Chapter 80: Install UD Agent Job

This section includes details about the Install UD Agent job.

Trigger Query



Job Parameters

By default, job parameters do not override adapter parameters.

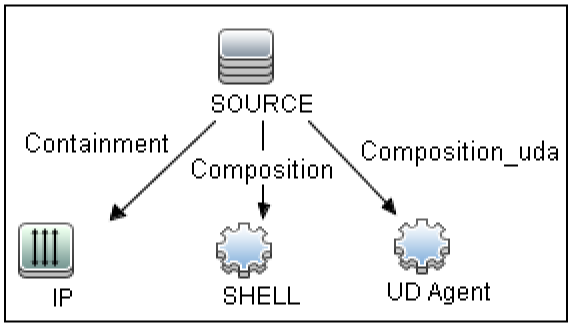
Adapter Information

This job uses the **InstallUDAgent** adapter.

Input CIT

Node

Input Query



Triggered CI Data

Name	Value
ProtocolList	${SHELL.root_class}
codepage	${SHELL.codepage:NA}
connected_os_credentials_id	${SHELL.connected_os_credentials_id:NA}
credentialsId	${SHELL.credentials_id}
hostId	${SOURCE.root_id}
ipTaggingList	${IP.ip_lease_time:NA}
ip_address	${SHELL.application_ip}
macList	${SHELL.arp_mac:NA}
nodeIpList	${IP.name:NA}
nodeMacList	${IP.arp_mac:NA}
shellId	${SHELL.root_id}

Discovered CITs

- Composition
- Node
- UDA

Adapter Parameters

Name	Default Value	Description
CallhomeFrequency	3	This parameter specifies the frequency of the CallHome request in days.
EnableSoftwareUtilization	false	This flag determines whether Software Utilization is enabled.

Name	Default Value	Description
PrimaryCallhomeProbeAddress		This parameter defines the primary Callhome Probe Address. Use one of the following formats: • IPAddress • HostName • HostNameOrIPv4Address:1977 • [IPv6Address]:1977 For example: • 10.11.12.13 • probehost • probehost:1977 • [2010:836b:4179::836b:4179]:1977
RunUDAgentUnderRootAccount	true	This parameter determines if the UD Agent runs under root account on Unix machines. The UD Agent runs under the installing user account if this parameter is false.
SecondaryCallhomeProbeAddress		This parameter specifies the secondary Callhome Probe Address. Use one of the formats as in PrimaryCallhomeProbeAddress.
SoftwareUtilizationPeriod	365	Software utilization data shows the number of days that an application was used (as a percentage) over the specified period.
UdAgentInstallCredentialId		This parameter specifies the UD Agent credential ID to be used to install the UD Agent. The installation process tries all IDs if this parameter is empty.

Global Configuration File

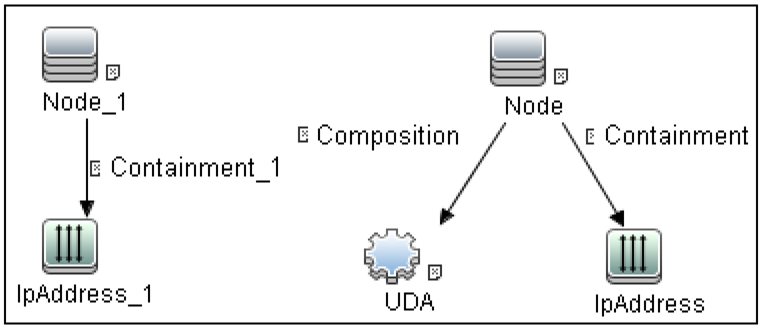
AgentsConfigurationByPlatform.xml

AgentsLifeCycleSettings.xml

Chapter 81: Migrate DDMI Agent Job

This section includes details about the Migrate DDMI Agent job.

Trigger Query



Job Parameters

By default, job parameters do not override adapter parameters.

Adapter Information

This job uses the **MigrateDDMItoUDAgentAdapter** adapter.

Input CIT

IpAddress

Input Query

None

Triggered CI Data

Name	Value
architecture	${UDA.architecture:NA}
codepage	${UDA.codepage:NA}

Name	Value
hostId	${SOURCE.root_id}
ip_address	${SOURCE.name}

Discovered CITs

- Composition
- Node
- UDA

Adapter Parameters

Name	Default Value	Description
BandwidthLimit	0	This parameter specifies the bandwidth limit to be applied while uploading or downloading in kb/s.
CallhomeFrequency	3	This parameter specifies the frequency of the CallHome request in days.
EnableSoftwareUtilization	false	This flag determines whether Software Utilization is enabled.

Name	Default Value	Description
PrimaryCallhomeProbeAddress		This parameter defines the primary Callhome Probe Address. Use one of the following formats: • IPAddress • HostName • HostNameOrIPv4Address:1977 • [IPv6Address]:1977 For example: • 10.11.12.13 • probehost • probehost:1977 • [2010:836b:4179::836b:4179]:1977
RunUDAgentUnderRootAccount	true	This parameter determines if the UD Agent runs under root account on Unix machines. The UD Agent runs under the installing user account if this parameter is false.
SecondaryCallhomeProbeAddress		This parameter specifies the secondary Callhome Probe Address. Use one of the formats as in PrimaryCallhomeProbeAddress.
SoftwareUtilizationPeriod	365	Software utilization data shows the number of days that an application was used (as a percentage) over the specified period.
codepage	NA	This parameter specifies the discovered machine code page.

Global Configuration File

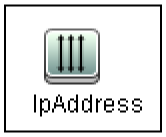
AgentsConfigurationByPlatform.xml

AgentsLifeCycleSettings.xml

Chapter 82: UDA Status Collector Job

This section includes details about the UDA Status Collector job.

Trigger Query



Job Parameters

By default, job parameters do not override adapter parameters.

Adapter Information

This job uses the **UDAStatusCollector** adapter.

Input CIT

IpAddress

Input Query

None

Triggered CI Data

Name	Value
domain_name	$(SOURCE.routing_domain:DefaultDomain)
ip_address	${SOURCE.name:NA}

Used Scripts

- icmp_utils.py

- UDAStatusCollector.py

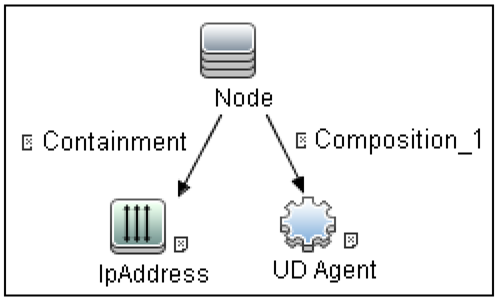
Adapter Parameters

Name	Default Value	Description
excludePatternsList		This parameter specifies a semicolon-separated (;) list of wildcard patterns. IP addresses that match any of the patterns are skipped. Pattern may include numbers, dots, * (matches zero or more characters) or ? (matches exactly one character).
isCreateUDA	false	This flag determines if the UDA CI is returned.
isIPv4PingEnabled	true	This flag determines whether to ping all IPv4 addresses within the defined ranges.
isIPv6PingEnabled	true	This flag determines whether to ping all IPv6 addresses within the defined ranges.
pingProtocol	1	This parameter specifies one of the PING protocol: • ICMP (1) • ECHO PORT (2) • Both ICMP and ECHO PORT (3)
range	NA	This parameter specifies a range of IPs to ping. Ranges are separated by semicolon. For example, 1.2.3.0-1.2.3.10;1.2.3.50-1.2.3.60
retryDiscover	2	This parameter specifies the retry times for an unsuccessful ping.
threadPoolSize	10	This parameter specifies the number of threads in pool that performs port 7 echoing.
timeoutDiscover	3000	This parameter specifies the ping time-out in ms.
virtualModeDiscover	false	This flag determines whether to discover the virtual IPs.

Chapter 83: Uninstall UD Agent Job

This section includes details about the Uninstall UD Agent job.

Trigger Query



Job Parameters

By default, job parameters do not override adapter parameters.

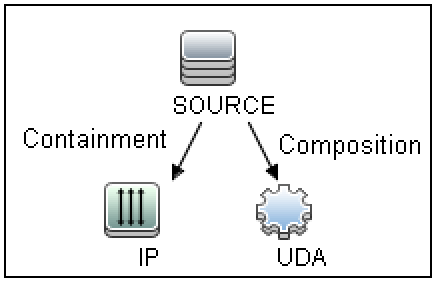
Adapter Information

This job uses the **UninstallUDAgent** adapter.

Input CIT

Node

Input Query



Triggered CI Data

Name	Value
ProtocolList	${UDA.root_class}
agentId	${UDA.root_id}
codepage	${UDA.codepage:NA}
connected_os_credentials_id	${UDA.connected_os_credentials_id:NA}
credentialsId	${UDA.credentials_id:NA}
hostId	${SOURCE.root_id}
ipTaggingList	${IP.ip_lease_time:NA}
ip_address	${UDA.application_ip}
macList	${UDA.arp_mac:NA}
nodeGUID	${SOURCE.ud_unique_id:NA}
nodeIpList	${IP.name:NA}
nodeMacList	${IP.arp_mac:NA}

Discovered CITs

- Composition
- NTCMD
- Node
- SSH
- Telnet

Adapter Parameters

Name	Default Value	Description
RemoveAgentData	false	This flag determines whether the Agent data is removed from the remote host after the Agent is uninstalled. For example, log file, utilization data.

Global Configuration File

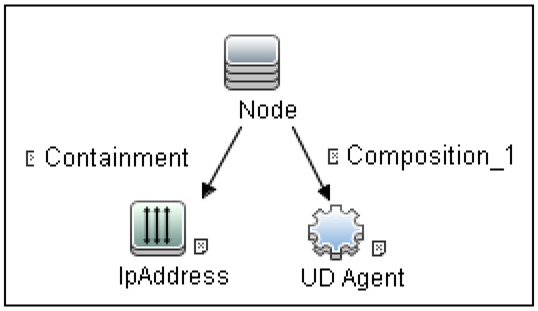
AgentsConfigurationByPlatform.xml

AgentsLifeCycleSettings.xml

Chapter 84: Update UD Agent Job

This section includes details about the Update UD Agent job.

Trigger Query



Job Parameters

By default, job parameters do not override adapter parameters.

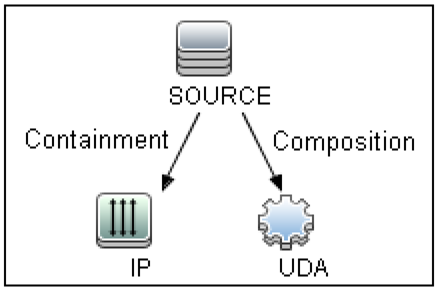
Adapter Information

This job uses the **UpdateUDAgent** adapter.

Input CIT

Node

Input Query



Triggered CI Data

Name	Value
ProtocolList	${UDA.root_class}
agentId	${UDA.root_id}
architecture	${UDA.architecture:NA}
codepage	${UDA.codepage:NA}
connected_os_credentials_id	${UDA.connected_os_credentials_id:NA}
credentialsId	${UDA.credentials_id:NA}
hostId	${SOURCE.root_id}
ipTaggingList	${IP.ip_lease_time:NA}
ip_address	${UDA.application_ip}
macList	${UDA.arp_mac:NA}
nodeIpList	${IP.name:NA}
nodeMacList	${IP.arp_mac:NA}
platform	${UDA.platform:NA}

Discovered CITs

- Composition
- Node
- UDA

Adapter Parameters

Name	Default Value	Description
CallhomeFrequency	3	This parameter specifies the frequency of the CallHome request in days.

Name	Default Value	Description
EnableSoftwareUtilization	false	This flag determines whether Software Utilization is enabled.
PrimaryCallhomeProbeAddress		This parameter defines the primary Callhome Probe Address. Use one of the following formats: • IPAddress • HostName • HostNameOrIPv4Address:1977 • [IPv6Address]:1977 For example: • 10.11.12.13 • probehost • probehost:1977 • [2010:836b:4179::836b:4179]:1977
RunUDAgentUnderRootAccount	true	This parameter determines if the UD Agent runs under root account on Unix machines. The UD Agent runs under the installing user account if this parameter is false.
SecondaryCallhomeProbeAddress		This parameter specifies the secondary Callhome Probe Address. Use one of the formats as in PrimaryCallhomeProbeAddress.
SoftwareUtilizationPeriod	365	Software utilization data shows the number of days that an application was used (as a percentage) over the specified period.
UdAgentInstallCredentialId		This parameter specifies the UD Agent credential ID to be used to install the UD Agent. The installation process tries all IDs if this parameter is empty.

Name	Default Value	Description
UpgradeAgent	true	This flag determines if the agent will be upgraded. When this parameter is false, only the agent configuration will be updated.

Global Configuration File

AgentsConfigurationByPlatform.xml

AgentsLifeCycleSettings.xml

Part 19: Top-Down Discovery

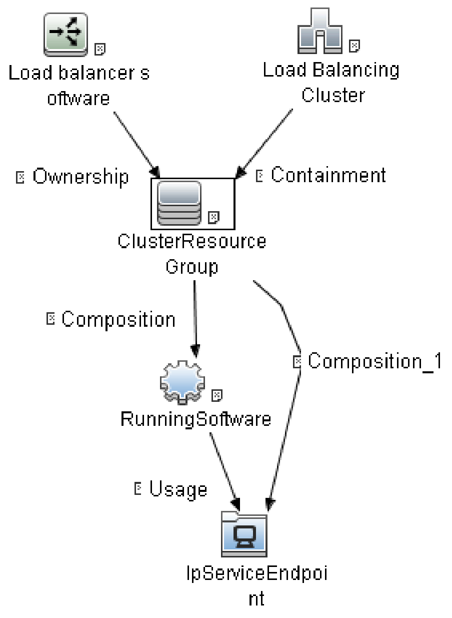
Chapter 85: F5 BIG-IP LTM Tunnel Job

This section includes details about the job.

Introduction

This job reports real IP addresses behind F5 to let top-down discovery continue if top-down URL contains a virtual IP address.

Trigger TQL



Node Name	Condition
Load balancer software	None
Load Balancing Cluster	None
ClusterResourceGroup	None
RunningSoftware	None
IpServiceEndpoint	None

Topology Map

None

Supported Policy

None

Adapter Information

This job uses the **F5 BIG-IP LTM Tunnel** adapter.

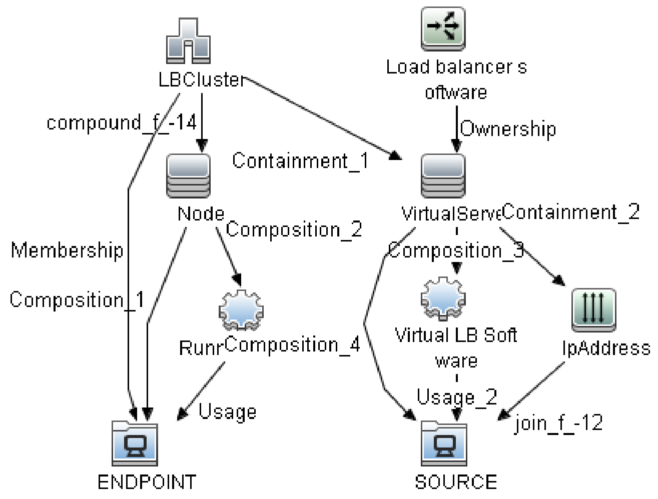
Adapter Type

Jython

Input CIT

IpServiceEndpoint

Input TQL



Triggered CI Data

Name	Value
REAL_IP_ADDRESS	${ENDPOINT.bound_to_ip_address}
VIRTUAL_IP_ADDRESS	${SOURCE.bound_to_ip_address}

Workflow Steps

None

Discovered CITs

IpAddress

Global Configuration Files

globalSettings.xml

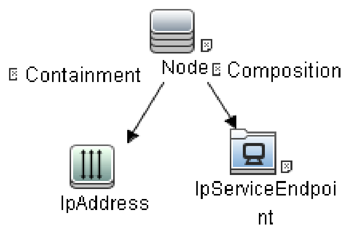
Chapter 86: JEE WebSphere Connections by JMX for Top-down Job

This section includes details about the job.

Introduction

This job discovers WebSphere servers based on either SOAP or RMI authentication.

Trigger TQL



Node Name	Condition
Node	None
IpAddress	None
IpServiceEndpoint	None

Topology Map

None

Supported Policy

None

Adapter Information

This job uses the **JEE WebSphere Connections by JMX for Top-down** adapter.

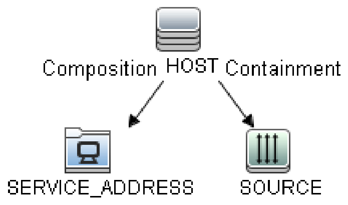
Adapter Type

Jython

Input CIT

IpAddress

Input TQL



Triggered CI Data

Name	Value
hostId	${HOST.root_id}
ip_address	${SOURCE.name}
ip_dnsname	${SOURCE.authoritative_dns_name:NA}
ip_domain	${SOURCE.routing_domain}
ports	${SERVICE_ADDRESS.network_port_number:NA}

Workflow Steps

None

Discovered CITs

- Composition
- IpAddress
- IpServiceEndpoint
- J2EE Domain
- JEE Node
- Node
- Usage
- WebSphere AS

Global Configuration Files

None

Chapter 87: Next-Hop Provider Job

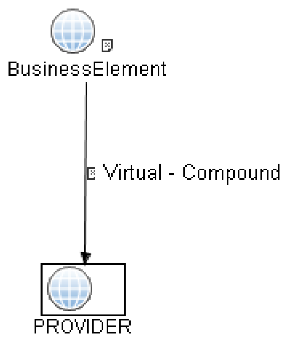
This section includes details about the job.

Introduction

This job discovers next-hop providers based on configuration files or TCP connections, excluding the following CI types: ISAMResources and subtypes, JMS Destination, ClusterResourceGroup, and RunningSoftware.

This job first extracts the provider-related information (IP address, DNS name, alias, or port) from the relevant configuration files of consumers. If configuration files are unavailable, this job will discover providers by capturing the opened TCP connections between consumers and providers.

Trigger TQL



Node Name	Condition
BusinessElement	Virtual - Compound (BusinessElement, PROVIDER) : 1..*

Node Name	Condition
PROVIDER	Virtual - Compound (BusinessElement, PROVIDER) : 1..* Excluded CI Types: • ISAMResources and subtypes • JMS Destination • ClusterResourceGroup • RunningSoftware

Topology Map

None

Supported Policy

None

Adapter Information

This job uses the **Next-Hop Provider** adapter.

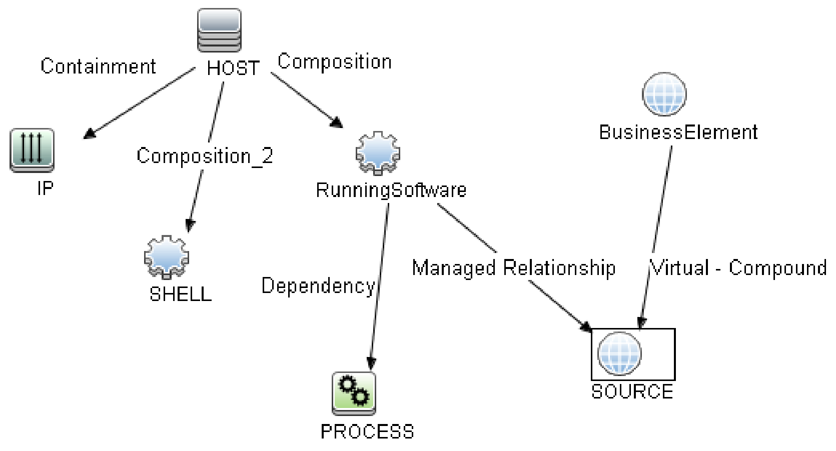
Adapter Type

Work Flow adapter

Input CIT

ConfigurationItem

Input TQL



Triggered CI Data

Name	Value
CONSUMERTYPE	${SOURCE.root_class}
Protocol	${SHELL.root_class:unknown}
codepage	${SHELL.codepage:NA}
connected_os_credentials_id	${SHELL.connected_os_credentials_id:NA}
credentialsId	${SHELL.credentials_id:NA}
hostId	${SHELL.root_container:NA}
host_ips	${IP.name}
ip_address	${SHELL.application_ip:unknown}
language	${SHELL.language:NA}
process_cmdline	${PROCESS.process_cmdline:NA}

Workflow Steps

Step Name	Module	Failure-policy

Default Search Result Awaiting	SendNextHopProviderResults.py	Mandatory
Next-Hop discovery by TCP Connection	tcp_discovery_server_asm.py	Mandatory

Discovered CITs

- IpAddress
- IpServiceEndpoint
- Node

Global Configuration Files

None

Chapter 88: Next-Hop Provider Job for Running Software

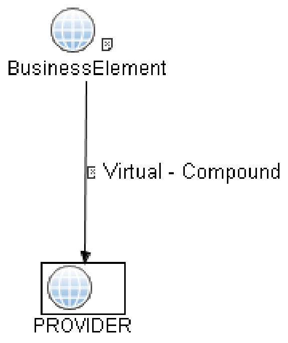
This section includes details about the job.

Introduction

This job discovers next-hop providers for RunningSoftware CIs based on configuration files or TCP connections.

This job first extracts the provider-related information (IP address, DNS name, alias, or port) from the relevant configuration files of consumers. If configuration files are unavailable, this job will discover providers by capturing the opened TCP connections between consumers and providers.

Trigger TQL



Node Name	Condition
BusinessElement	Virtual - Compound (BusinessElement, PROVIDER) : 1..*
PROVIDER	Virtual - Compound (BusinessElement, PROVIDER) : 1..* Included CI Types: RunningSoftware and subtypes

Topology Map

None

Supported Policy

None

Adapter Information

This job uses the **Next-Hop Provider for Running Software** adapter.

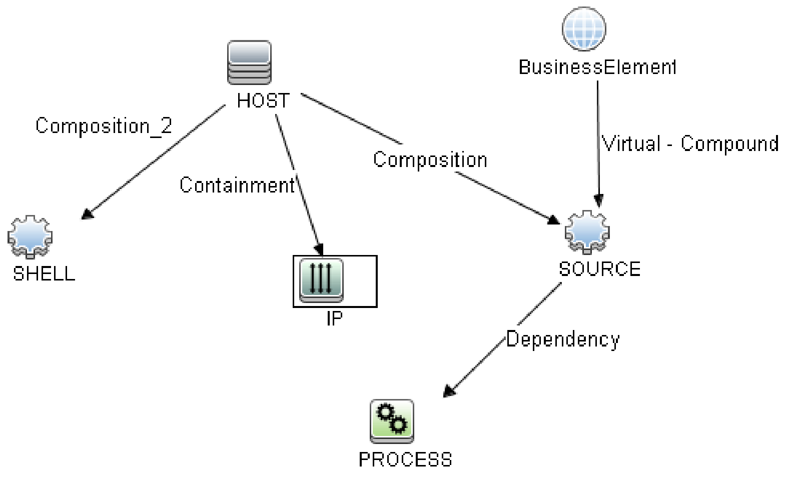
Adapter Type

Work Flow adapter

Input CIT

ConfigurationItem

Input TQL



Triggered CI Data

Name	Value
CONSUMERTYPE	${SOURCE.root_class}
Protocol	${SHELL.root_class:unknown}
codepage	${SHELL.codepage:NA}
connected_os_credentials_id	${SHELL.connected_os_credentials_id:NA}
credentialsId	${SHELL.credentials_id:NA}
hostId	${SHELL.root_container:NA}
host_ips	${IP.name}
ip_address	${SHELL.application_ip:unknown}
language	${SHELL.language:NA}
process_cmdline	${PROCESS.process_cmdline:NA}

Workflow Steps

Step Name	Module	Failure-policy
Default Search Result Awaiting	SendNextHopProviderResults.py	Mandatory
Next-Hop discovery by TCP Connection	tcp_discovery_server_asm.py	Mandatory

Discovered CITs

- IpAddress
- IpServiceEndpoint
- Node

Global Configuration Files

None

Chapter 89: URL Resolver

This chapter includes:

Overview

This adapter outputs node information, IP addresses, and running software information from an HTTP URL string.

Introduction

The adapter resolves the URL string value of the attribute "Description" in BusinessElement CI and then writes the string value to a text plain file named url.txt.

Prerequisites

Before running the URL Resolver job, you must complete the following:

Go to **Adapter Management** > **Resources** pane > click **URL Resolver** > **Adapter Configuration** tab. Ensure that you select **Override default Probe selection**, and that you type the name or IP address of the Data Flow Probe that you want to run the job.

Adapter

Trigger TQL

BusinessElement

Input CIT

BusinessElement

Triggered CI Data

Name	Value
url	${SOURCE.description}

Used Scripts

- URL_Resover.py
- Discovered CITs
- ConfigurationDocument
- IpAddress
- UriEndpoint
- Node

- Containment
- Composition
- Usage
- Dependency

Global Configuration Files

globalSettings.xml

Parameters

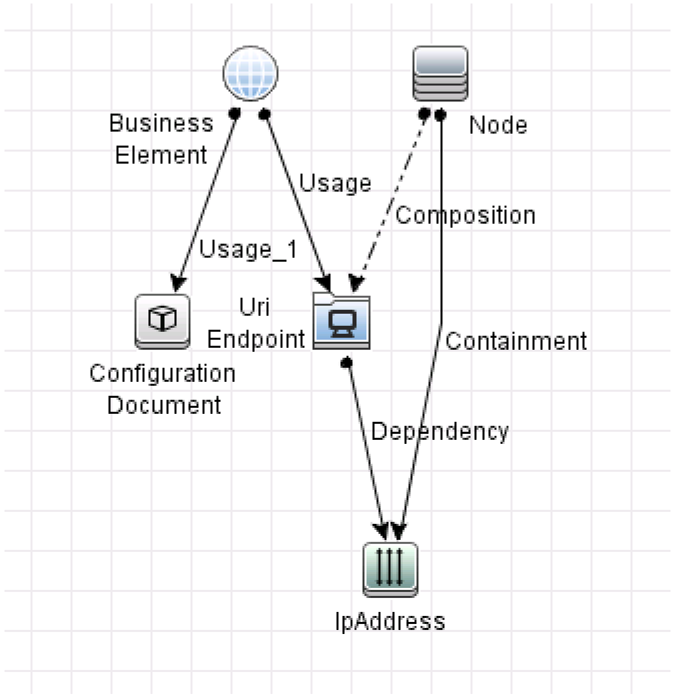
dnsServers

Supported Protocols

The adapter supports HTTP/HTTPS URL string.

Topology

The following image displays the topology of this adapter.


Business Element
Node
Usage
Composition
Usage_1
Uri Endpoint
Containment
Configuration Document
Dependency
IpAddress

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Discovery and Integrations Content Guide - Discovery Modules (Universal CMDB Content Pack 15.00 (CP15))

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to cms-doc@hp.com.

We appreciate your feedback!