HP Universal CMDB

For the Windows and Red Hat Enterprise Linux operating systems

Software Version: 9.05, CP 11.00

HP UCMDB Discovery and Integration Content Guide

Document Release Date: June 2012

Software Release Date: June 2012

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

Java and Oracle are registered trademarks of Oracle Corporation and/or its affiliates.

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Acknowledgements

- This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
- This product includes OpenLDAP code from OpenLDAP Foundation (http://www.openldap.org/foundation/).
- This product includes GNU code from Free Software Foundation, Inc. (http://www.fsf.org/).
- This product includes JiBX code from Dennis M. Sosnoski.
- This product includes the XPP3 XMLPull parser included in the distribution and used throughout JiBX, from Extreme! Lab, Indiana University.
- This product includes the Office Look and Feels License from Robert Futrell (http://sourceforge.net/projects/officelnfs).
- This product includes JEP - Java Expression Parser code from Netaphor Software, Inc. (http://www.netaphor.com/home.asp).

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:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link 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 Online web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site 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 web site 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 sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Part I: General Reference and Supported Content

Chapter 1

General Reference

This chapter includes:

How to Define a New Port

You define a new port by editing the portNumberToPortName.xml file:

1. In the Adapter Management window (**Admin > RTSM Administration > Data Flow Management > Adapter Management**), search for the `portNumberToPortName.xml` file: click the **Find resource** button and enter **portNumberToPortName.xml** in the **Name** box. Click **Find Next**, then click **Close**.

   The file is selected in the Resources pane and the file contents are displayed in the View pane.

   For details about this file, see "portNumberToPortName.xml File" on page 64.

2. Add another row to the file and make changes to the parameters:

   ```
   <portInfo portProtocol="xxx" portNumber="xxx" portName="xxx"
   discover="0" cpVersion="xx"/>
   ```

Parameter	Description
portProtocol	The network protocol used for discovery (`udp` or `tcp`).
portNumber	The port number to be discovered. This attribute may be a number or a range. Ranges may be separated by commas or dashes or both. For example: "10, 21, 45", "10-21", or "10-21, 45, 110".
portName	The name that is to be displayed for this port.
discover	**1**. This port must be discovered. **0**: This port should not be discovered.

cpVersion	Use this parameter when you want to export the **portNumberToPortName.xml** file to another UCMDB system with the Package Manager. If the **portNumberToPortName.xml** file on the other system includes ports for this application but does not include the new port you want to add, the **cpVersion** attribute ensures that the new port information is copied to the file on the other system.
	The **cpVersion** value must be greater than the value that appears in the root of the **portNumberToPortName.xml** file.
	For example, if the root **cpVersion** value is **3**:
	`<portList parserClassName="com.hp.ucmdb.discovery.library.communication.downloader.cfgfiles.KnownPortsConfigFile" cpVersion="3">`
	the new port entry must include a **cpVersion** value of **4**:
	`<portInfo portProtocol="udp" portNumber="1" portName="A1" discover="0" cpVersion="4"/>`

Parameter	Description
	Note: If the root **cpVersion** value is missing, you can add any non-negative number to the new port entry. This parameter is also needed during Content Pack upgrade. For details, see "How to Use the cpVersion Attribute to Verify Content Update" on next page.

How to Use the cpVersion Attribute to Verify Content Update

The **cpVersion** attribute is included in the portNumberToPortName.xml file, and indicates in which Content Pack release a port has been discovered. For example, the following code defines that the LDAP port 389 has been discovered in Content Pack 11.00:

```
<portInfo portProtocol="tcp" portNumber="389" portName="ldap"
discover="11" cpVersion="11"/>
```

During a Content Pack upgrade, DFM uses this attribute to perform a smart merge between the existing portNumberToPortName.xml file (which may include user-defined ports) and the new file. Entries previously added by the user are not removed and entries previously deleted by the user are not added.

For details about the **portNumberToPortName.xml** file, see "portNumberToPortName.xml File" on page 64.

To verify that a Content Pack is successfully deployed:

1. Install the latest Service Pack release.
2. Start the UCMDB Server.
3. Verify that all services are running. For details, see the section about HP Universal CMDB Services in the interactive *HP Universal CMDB Deployment Guide*.
4. Install and deploy the latest Content Pack release. For details, refer to the Content Pack installation guide.
5. In the Adapter Management window, access the **portNumberToPortName.xml** file.
6. Verify that no user-defined ports have been deleted and that any ports deleted by the user have not been added.

How to Delete Files Copied to Remote Machine

During discovery, the Data Flow Probe copies files to a remote Windows machine. For details, see "Files Copied to a Remote Machine" on page 51.

To configure DFM to delete files copied to the destination machine after discovery is finished:

1. Access the **globalSettings.xml** file: **Adapter Management > AutoDiscoveryContent > Configuration Files**.
2. Locate the **removeCopiedFiles** parameter.
   - **true**. The files are deleted.
   - **false**. The files are not deleted.
3. Save the file.

To control xCmd behavior:

1. In the **globalSettings.xml** file, locate the **NtcmdAgentRetention** parameter.
2. Enter one of the following:
   - **0**. (The default) Unregister the service and delete the remote executable file. (**Unregister**: stop the service and remove it from the remote machine, so that it is no longer listed in the list of services.)
   - **1**. Unregister the service, but leave the executable file on the file system.
   - **2**. Leave the service running, and leave the executable file on the file system.

How to Run xCmd from a Windows 2008/R2 Machine

Perform the following to ensure that xCmd functions properly when the Probe is installed on a Windows 2008/R2 machine:

1. Stop the Probe.
2. Open the standard Windows Registry Editor application by running the **regedit** executable.
3. In the Registry Editor navigate to the following registry key:

   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control
4. Under this key there should be a **REG_DWORD** parameter **SCMApiConnectionParam**
   a. If this is missing, add a new **REG_DWORD** parameter **SCMApiConnectionParam** and set its value to 0x80000000.
   b. If this value is already available in the registry, combine it with the 0x80000000 mask (using bitwise OR). For example, if there was a value 0x1 in there, you need to set this value to 0x80000001.

Note: To run xCmd from a Windows 2008/R2 machine **with UAC enabled**, also perform the following additional steps.

1. Stop the Probe.
2. Locate the **xCmd.exe** file in the **hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources** directory.
3. Right-click the **xCmd.exe** file and select **Properties**.
4. In the **Compatibility** tab:
   a. Select **Compatibility mode**.
   b. Select **Run this program in compatibility for**: **Windows XP (Service Pack 2)**.
   c. Select **Run this program as administrator**.
5. Locate the **wrapper.exe** file, in the hp\UCMDB\DataFlowProbe\bin directory.
6. Right-click the **wrapper.exe** file, and select **Properties**.
7. In the **Compatibility** tab:
   a. Select **Compatibility mode**.
   b. Select **Run this program in compatibility for**: **Windows XP (Service Pack 2)**.
   c. Select **Run this program as administrator**.
8. Start the Probe.

Note: xCmd uses DCOM protocol for connecting to remote machines.

The DCOM protocol requires that the following ports are open: **135**, **137**, **138,** and **139**.

In addition it uses arbitrary ports between **1024** and **65535**, but there are ways to restrict the port range used by WMI/DCOM/RPC.

For information about configuring DCOM to work with firewalls, see http://support.microsoft.com/kb/154596/en-us.

Files Copied to a Remote Machine

During discovery, Data Flow Probe copies files to a remote Windows machine to enable discovery of the machine's components. The files are copied to the **%SystemRoot%\system32\drivers\etc** folder on the remote machine.

Note:

- Data Flow Management runs **xCmdSvc.exe** to connect to and retrieve the Shell on the remote machine.
- When the **wmic** command is launched on the remote Windows machine, by the **Host Connection by Shell** or **Host Resources and Applications by Shell** jobs, an empty **TempWmicBatchFile.bat** file is created.

The following files are copied:

File	Content Pack Version	Description
adsutil.vbs	All	The Visual Basic script used for discovery of Microsoft IIS applications. DFM copies this script to the remote machine to discover IIS. **Relevant DFM Job:** IIS Applications by NTCMD or UDA
diskinfo.exe	All	The executable that enables the retrieval of disk information when it is not available to be retrieved by **wmic**. DFM discovers default disk information with the **wmic** query. However, if the **wmic** query fails to execute, DFM copies the **diskinfo.exe** file to the remote machine. This failure can occur if, for example **wmic.exe** is not included in the PATH system variable or is completely absent on the remote machine, as is the case on Windows 2000. **Relevant DFM Job:** Host Resources and Applications by Shell

File	Content Pack Version	Description
Exchange_Server_2007_Discovery.ps1	CP4	The PowerShell script for MS Exchange 2007 discovery. DFM uses a PowerShell scenario to discover Microsoft Exchange 2007 by NTCMD. This file, therefore, must be copied to the remote machine. **Relevant DFM Jobs:** • Microsoft Exchange Connection by NTCMD or UDA • Microsoft Exchange Topology by NTCMD or UDA
GetFileModificationDate.vbs	CP5	The Visual Basic script for retrieving the file modification date (disregarding locale). The most common use case is when DFM must retrieve the last modification date of a configuration file of a discovered application. **Relevant DFM Jobs:** • Apache Tomcat by Shell • File Monitor by Shell • IIS Applications by NTCMD or UDA • J2EE Weblogic by Shell • J2EE WebSphere by Shell or JMX • J2EE WebSphere by Shell • Oracle TNSName by Shell • SAP Profiles by Shell • SAP System By Shell • Service Guard Cluster Topology by TTY • Siebel Application Server Configuration • Software Element CF by Shell • Veritas Cluster by Shell • Webserver by Shell

File	Content Pack Version	Description
getfilever.vbs	All	The Visual Basic script used to identify the version of the running software. The script retrieves the executable or DLL file version on Windows machines. This script is used by Shell-based application signature plug-ins to retrieve the version of a particular software on the remote machine. **Relevant DFM Job:** Host Resources and Applications by Shell
junction.exe	CP5	This executable file, part of the Sysinternals Suite (http://technet.microsoft.com/en-us/sysinternals/bb842062.aspx), enables the creation of a junction point. DFM uses this file if the **linkd.exe** and **mklink.exe** tools are absent on the remote machine. When DFM runs discovery on a Windows x64 machine, DFM needs to bypass the Windows redirect feature running on that machine. DFM does this by creating a link to the **%SystemRoot%\System32** folder with either the **linkd.exe** or **mklink.exe** tool. However, if these tools are missing on the remote machine, DFM transfers **junction.exe** to the remote machine. DFM is then able to launch the 64-bit version of the system executable files. (Without this 64-bit version, DFM would be locked into an isolated 32-bit world.) This junction point is automatically removed once discovery is complete. **Relevant DFM Jobs:** • Host Resources and Applications by Shell • Microsoft Exchange Connection by NTCMD or UDA • Microsoft Exchange Topology by NTCMD or UDA

File	Content Pack Version	Description
meminfo.exe	All	The executable that enables the retrieval of memory information. DFM discovers memory information with the **wmic** query. However, if the **wmic** query fails to execute, DFM copies the **meminfo.exe** file to the remote machine. This failure can occur if, for example, **wmic.exe** is not included in the PATH system variable or is completely absent on the remote machine, as is the case on Windows 2000. **Relevant DFM Job:** Host Resources and Applications by Shell
reg_mam.exe	All	The copy of the Microsoft reg.exe file that enables querying the registry. If DFM does not discover a native reg.exe file, this executable is copied to the remote Windows machine. This situation occurs with some previous Windows versions (for example, Windows 2000) where the tool is not included by default but can still function there correctly. **Relevant DFM Job:** Host Resources and Applications by Shell

Content Pack Configuration Files

The Content Pack contains configuration files which enable you to configure commonly used parameters such as command timeouts, usage of some utilities, application signatures, and so on.

This section includes:

globalSettings.xml File

The following table describes the parameters in the **globalSettings.xml** configuration file:

Parameter	Description
AdditionalClasspath	Additional path that enables to run different patterns (i.e. database patterns); all paths should be relative to the **$PROBE_INSTALL/root/lib/collectors/probeManager/discoveryResources/** folder and should be semicolon separated **Example:** **<property name="AdditionalClasspath"> db/oracle/.;db/mssqlserver/.</property>** means that following paths will be included in the classpath: • **$PROBE_INSTALL/root/lib/collectors/probeManager/discoveryResources/db/oracle/** • **$PROBE_INSTALL/root/lib/collectors/probeManager/discoveryResources/db/mssqlserver/**
allowGettingCredentialSecuredAttribute	Indicates whether Jython scripts are allowed to get credentials secured data (true) or not (false). If this setting is set to false, then Jython scripts are not allowed to retrieve sensitive credentials data (like passwords that are stored on the server side). **Default:** true
autoTruncateDbEncoding	Indicates the encoding used by the CMDB underlaying database. This property is used during results truncation property (in case the property was identified as auto-truncate enabled) for calculating number of characters that should be sent after truncation. **Default:** UTF8

Parameter	Description
autoTruncatePercentage	If the value of the attribute (with the DDM_AUTOTRUNCATE qualifier) exceeds the size limit multiplied by this parameter it will be truncated to the specified part of the defined size. **Default:** 100 percent
clearCommandLineFor Processes	Clears the Command line for these processes. This option is used to ensure that no private or confidential data is stored in CMDB. **Default:** xCmd.exe, srvrmgr.exe, srvrmgr. **Syntax exceptions:** Process names are case insensitive and should be split by commas.
dbQueryTimeout	The timeout (in seconds) for all SQL queries. Indicates how long to wait for query results. The timeout applies only if the value is greater than zero (0). **Default:** 100 seconds **Note:** Some JDBC drivers can not support this setting.
defaultSapClients	When this parameter is defined, you do not need to specify the SAP Client Number parameter in the SAP ABAP protocol. Instead, you can create one or more comma-separated credentials for multiple SAP systems with different supported clients. **Example:** `<property name= "defaultSapClients"> 800,500,200,300 </property>` **Default:** 800
desktopOperating Systems serverOperating Systems	These two parameters are used to determine if the host's operating system is of type Desktop or Server. If the host's operating system name contains a value from one of these lists, its **host_isdesktop** is set accordingly. Otherwise the value of **host_ isdesktop** attribute is left empty.
discovereAllListenPorts	Related to application signature configuration.

Parameter	Description
discoveredStorageTypes	Describes storage types which have to be reported to UCMDB. Options are split by commas. Available options are: • FixedDisk • NetworkDisk • CompactDisk • RemovableDisk • FloppyDisk • VirtualMemory • FlashMemory • RamDisk • Ram • No Root Directory • Other • UNKNOWN
ignoreLocalized VirtualInterfaces PatternList	Lists patterns for localized Windows Virtual interface description that must not take part in the Host Key creation process. **Format:** Comma-separated list of strings, no additional white-spaces allowed.
ignoreVmwareInterfaces	Indicates whether to ignore the VMware MAC address. • **When there is a Physical MAC** (default). The VMware MAC address is used only if the pattern can not find any physical MAC address. • **Always.** Always ignore VMware MAC address.

Parameter	Description

jdbcDrivers	This section enumerates driver classes used to connect to a dedicated Database server. Names of sub-keys must be the same as used in credentials (sqlprotocol_dbtype attribute of protocol).
	
	Change them if drivers other than OOTB JDBC drivers are used.
	
	Default values for OOTB-installation:
	
	`<property name="jdbcDrivers:>`
	`<oracle>`
	`com.inet.ora.OraDriver`
	`</oracle>`
	`<oracleSSL>`
	`com.mercury.`
	`jdbc.oracle.`
	`OracleDriver`
	`</oracleSSL>`
	`<MicrosoftSQLServer>`
	`net.sourceforge.`
	`jtds.jdbc.Driver`
	`</MicrosoftSQLServer>`
	`<MicrosoftSQLServer> net.sourceforge.jtds.`
	`jdbc.Driver`
	`</MicrosoftSQLServerNTLM>`
	`<MicrosoftSQLServerNTLMv2>`
	`net.sourceforge.jtds.jdbc.Driver`
	`</MicrosoftSQLServerNTLMv2>`
	`<Sybase>`
	`com.sybase.jdbc.SybDriver`
	`</Sybase>`
	`<db2>`
	`com.ibm.db2.jcc.DB2Driver`
	`</db2>`
	`<mysql>`
	`com.mysql.jdbc.Driver`
	`</mysql>`
	`</property>`

Parameter	Description
jdbcPreUrls	This section enumerates URL templates used to connect to dedicated Database server. Names of sub-keys must be the same as those used in credentials (sqlprotocol_dbtype attribute of protocol). Change them if drivers other than OOTB JDBC drivers are used. Values depend on used drivers and should be taken from driver documentation. **Note:** Symbol ampersand (&) must be escaped according to XML standard (&) **Default values for OOTB-installation:** <property name="jdbcPreUrls"> <oracle> jdbc:inetora:%%ipaddress%%: %%protocol_port%%: %%sqlprotocol_dbsid %%?logging=false&loginTimeout =%%protocol_timeout%% </oracle> <oracleSSL> jdbc:mercury:oracle:// %%ipaddress%%:%%protocol_port%%; ServiceName= %%sqlprotocol_dbsid%% </oracleSSL> <MicrosoftSQLServer> jdbc:jtds:sqlserver:// %%ipaddress%%:%%protocol_port%%; instanceName=%%sqlprotocol_dbname%%; loginTimeout=%%protocol_timeout%%; logging=false;ssl=request </MicrosoftSQLServer>

Parameter	Description
jdbcPreUrls continued	<MicrosoftSQLServerNTLM> jdbc:jtds: sqlserver://%%ipaddress%%: %%protocol_port%%;instanceName= %%sqlprotocol_dbname%%;domain= %%sqlprotocol_windomain%%; loginTimeout= %%protocol_timeout%%;logging=false </MicrosoftSQLServerNTLM> <MicrosoftSQLServerNTLMv2>jdbc:jtds:sqlserver://%% ipaddress%%:%%protocol_port%%;instanceName=%% sqlprotocol_dbname%%;domain=%% sqlprotocol_windomain%%;loginTimeout=%% protocol_ timeout%%;logging=false;ssl=request;useNTLMv2=true </MicrosoftSQLServerNTLMv2> <Sybase> jdbc:sybase:Tds: %%ipaddress%% :%%protocol_port%%?DatabaseName= %%sqlprotocol_dbname%% </Sybase> <db2> jdbc:db2://%%ipaddress%%: %%protocol_port%%/ %%sqlprotocol_dbname%% </db2> <mysql> jdbc:mysql://%%ipaddress%%: %%protocol_port%%/ %%sqlprotocol_dbname %%</mysql> </property>
loadExternalDTD	Used to configure file_mon_utils to prevent downloading DTD files while validating the XML. **Default:** false
maxExecutionRecords	Specifies maximal number of execution records that can be in the communication log. This parameter should be used when the discovery process discovers a lot of data. The parameter can be overridden on an adapter level. In this case, add the parameter to the adapter with desired record limit (see Probe documentation). **Default:** -1 means unlimited

Parameter	Description
maxStoreSentResults	Specifies maximal number of sent results that can be stored in the communication log. This parameter can be changed if there are too many results stored in the communication log. If this value is greater than 0, the log will store the corresponding number of results for deleted results AND updated results, meaning that the results set will contain double the value of **maxStoreSentResults.** **Default:** -1 means unlimited
multipleUpdateIgnore Types	Used by UCMDB. The Probe does not generate a **Multiple updates in bulk** warning for enumerated CI Types.
NtcmdAgentRetention	NTCMD agent retention mode. Specifies how to handle a remote NTCMD service and its executable file when closing the connection. • **0** (default)**.** Unregister the service and delete the remote executable file. • **1.** Unregister the service but keep the executable file on the file system. • **2.** Leave the service running, keep the executable file.
NtcmdSessionUse ProcessBuilder	This parameter is for **NtcmdSessionAgent** and should be always be **true**. This parameter tells how to create a new process. • **true.** The new process will be created by ProcessBuilder (new API from Java 5.0) • **false.** The new process will be created by Runtime.exec (old API, from Java 1.4.2). Set to false only in case of backward compatibility problems.
objectSendAmount Threshold	When the number of discovered objects exceeds this threshold, the objects are immediately sent to the server. Requires using the sendObject(s) API in jython scripts. **Default:** 2000 objects
objectSendTime Threshold	When more than the specified time (in seconds) has passed since the previous object report, the objects are immediately sent to the server. Requires using the 0sendObject(s) API in jython scripts. **Default**: 300 seconds
portExpirationTime	The expiration time (in seconds) of the TCP/UDP port entry in the Probe's database. **Default:** 60 seconds

Parameter	Description
powershellConnection IdleTimeout	Defines the maximum idle time (in milliseconds) for the powershellconnector.exe process. The timer resets its state after each command execution. **Default:** 3600000 milliseconds (1h)
processExpirationTime	The expiration time (in seconds) of the Process entry in the Probe database. **Default:** 60 seconds
remoteProcessTimeout	After being launched, the remote process should connect with the Probe within the defined time (in milliseconds), otherwise the following error is produced: **Failed to connect to remote process**. **Default:** 300000 milliseconds (5 minutes)
removeCopiedFiles	In some cases DFM copies scripts and third-party utilities on a client machine. The **removeCopiedFiles** parameter defines whether these files should (true) or should not (false) be deleted after discovery is finished.
ResultProcessIsLenient	When set to **true,** the discovery result processing is lenient (not recommended): • If a reported string attribute has too large a value, the string it is automatically truncated according to the CMDB Class Model definition • If the OSH attribute is invalid (type/nonexisting attribute/missing ID attribute) only the invalid OSH is dropped, rather than entire bulk (default)
setBiosUuidTo MicrosoftStandart	Indicates whether the BIOS UUID value for Windows operating systems should be reported in Microsoft style (some bytes order reversed) instead of the original BIOS value. Affects Host Connection jobs. • **false**. Converts to original BIOS stored value • **true**. Converts to Microsoft standard. **Note:** Setting this parameter to **true** may result in conflicts with the BIOS UUID value discovered by VMware jobs or some integrations.
shellGlobal CommandTimeout	Global timeout (in milliseconds) for all Shell client commands. Indicates how long to wait for a command's result. **Default:** 15000 milliseconds
siebelCommandTimeout	The amount of time to wait for the Siebel command's result. **Default:** 3 minutes (180000 ms)

Parameter	Description
snmpGlobalRequestTimeout	This is the time, in milliseconds, after which a request using SNMP will timeout. **Default:** 3,000 milliseconds **Note:** This value is global for all SNMP requests. If you want to override the SNMP request timeout for a specific query (where you know the query takes more time than the default timeout), provide the timeout value as a second parameter to the executeQuery method on the SNMP client: **snmpClient.executeQuery(SNMP_QUERY_STRING, QUERY_TIMEOUT_IN_MILLISECONDS).**

snmpTestQueries	Defines the default SNMP test query for SNMP Agent. Can be overridden for specific devices.
	
	Default:
	
	`<property name="snmpTestQueries">`
	
	`<query>`
	`1.3.6.1.2.1.1.1,1.3.6.1.2.1.1.2,`
	`string</query>`
	
	`</property>`

tcpExpirationTime	The expiration time (in hours) of TCP connection entry in probe database. **Default:** 24 hours

tnsnamesFilePaths	Paths to search the **tnsnames.ora** file (including **tnsnames.ora** itself, comma separated)
	
	Example:
	
	`<property name=`
	`"tnsnamesFilePaths">`
	`c:\temp\tnsnames.ora`
	`</property>`

useIntermediate FileForWmic	Usage of an intermediate temporary file for data transfer by wmic command. **Default:** false
useJinteropOnLinux	This setting is used on non-Windows machines and • **true** (default)**.** The Probe uses JInterop for WMI discovery. • **false.** The Probe uses Windows remote Proxy.

Parameter	Description
useJinteropOnWindows	This property is used on Windows machines. • **true.** The Probe uses JInterop for WMI discovery. • **false** (default)**.** The Probe uses WMIdll native code.
useNtcmdModified Markers	• **true.** The Probe uses markers with counters in NTCMD agents' infrastructure. • **false.** The Probe uses old NTCMD behavior - without markers with counters.
useSnmp4j	Affects jobs * by SNMP. Defines which SNMP library to use for SNMP queries. • **true** (default). SNMP4J library are used. • **false.** Inner implementations are used.
useWinexeOnLinux	This setting is used on non-Windows machines. • **true.** The Probe uses local winexe executable for NTCMD Windows discovery. • **false** (default)**.** The Probe uses Windows remote Proxy.

portNumberToPortName.xml File

The **portNumberToPortName.xml** file is used by DFM as a dictionary to create IpServiceEndpoint CIs by mapping port numbers to meaningful port names. When a port is discovered, the Probe extracts the port number, searches in the **portNumberToPortName.xml** file for the port name that corresponds to this port number, and creates the IpServiceEndpoint CI with that name. If the port name does not appear in this file, the Probe uses the port number as the port name.

You can specify different names for same port number for different IP ranges. In this case, the same port discovered for IPs contained in different ranges will have different port names.

Note: The **portNumber** attribute may be a number or a range. Ranges may be separated by commas or dashes or both. For example: "10, 21, 45", "10-21", or "10-21, 45, 110".

For details on adding new ports to be discovered, see "How to Define a New Port" on page 46.

Troubleshooting and Limitations

This section describes general troubleshooting and limitation related to performing discovery using Universal Discovery.

- **Problem**: Cannot Connect to Windows Vista/2008-R2 Machines with UAC Enabled

  Reason: Starting from Windows Vista, Microsoft has changed the security mechanism by

introducing the UAC (User Account Control) technology. This change causes problems with xCmd connecting to remote Windows Vista/2008-R2 machines when using the local administrator account.

Solution: The following procedure enables xCmd connection to remote Windows Vista/2008-R2 machines with UAC enabled.

a. Verify the xCmd connection

   i. Log in to the Probe machine.

   ii. Locate the **xcmd.exe** file in hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources directory.

   iii. Open **cmd.com** in the same directory.

   iv. At the command prompt, invoke following command:

```
xCmd.exe \\ <problematic machine name or ip>
//USER:<domain>\<username> cmd
```

   v. Enter the required password.

b. If the xCmd connection is not successful, check accessibility to the shared folder, admin$.

   Ensure that the Probe machine can access the shared folder, **admin$**, on the remote machine.

   i. Log in to the Probe machine.

   ii. Select **Start > Run**, and enter `\\<remote machine>\admin$ address`.

   iii. If there is no access to **admin$**:

   - Log in to the remote machine.
   - Select **Start > Run**, and enter `regedit`.
   - Locate the following registry subkey:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
LanmanServer\Parameters
```

   - Right-click **Parameters**, and select the **Details** pane.
   - If the **AutoShareServer** registry entry does not exist, in the **Edit** menu, select **New > DWORD (32-bit) Value.** Enter **AutoShareServer**, and click **OK**.
   - Select **AutoShareServer**. In the **Edit** menu, select **Modify**, and in the **Value** box, type `1`.
   - Exit the Registry Editor, and restart the computer.
   - Select **Start > Run**, and enter `net start srvnet`.

iv. When access to **admin$** is successful, try to verify the xCmd connection again as described in "Verify the xCmd connection" on previous page.

c. If the verification still fails, connect to Windows Vista/2008-R2 machines with UAC enabled.

i. On Windows Vista/2008-R2 machines, local administrators do not have full privileges when connected remotely.

Use one of the following options to overcome this problem:

- Connect using domain administrator credentials.
- Enable local administrators to have full privileges by modifying the registry on remote machine as follows:

Key	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ Windows\CurrentVersion\Policies\system
Value	LocalAccountTokenFilterPolicy should be set to 1. If this value is not available, create a new DWORD value and set it to 1.

ii. Restart the machine.

- **Problem:**The file transfer does not work when communicating with the remote Linux/UNIX/Mac OS X machines, as the result operations like Scanner-based Inventory Discovery or deployment of Universal Discovery agents fail.

  Solution:

  a. Make sure the SSH agent is configured to allow file transfer via the SCP/SFTP protocols.

  b. Make sure that the logon process for the user that is used for the SSH protocol does not have a banner that requires manual user input during the logon process.

Chapter 2

Supported Content

This chapter includes:

Discovered Applications

Note: Additional supported content is publicly available to download through the HP Live Network (https://hpln.hp.com). Follow the **Discovery and Dependency Mapping** quick link. You will need an HP Passport user name and password.

Vendor	Product	Versions	Credentials	Discovers...
Amazon	Amazon Web Services		AWS	EC2 and RDS topologies.
Apache	Http Server	1.3, 2.0, 2.2	Shell	Apache Http server Listening ports, Virtual hosts, configuration files, Web application, Apache Modules (including mod_proxy and mod_ proxy_balancer.
Apache	Tomcat	5, 5.5, 6.0	Shell	Tomcat Server, Web applications, configuration files, virtual servers, listening ports, Tomcat Cluster, Tomcat Service.
BMC	Atrium CMDB	1.1, 2.0, 2.1, 7.5, 7.6	Remedy	Pushes configuration items (CIs) from HP UCMDB to the Atrium CMDB server using mapping xml files. **Note**: Synchronized Content, not discovery of application topology.
BMC	Remedy ARS	6.3, 7.0, 7.1, 7.5, 7.6	Remedy	Pushes CIs from HP UCMDB to Remedy ARS using mapping xml files. **Note**: Synchronized Content, not discovery of application topology.
CA Technologies	CA CMDB	12.0, 12.5	CA CMDB protocol	Pushes CIs from HP UCMDB to the CA CMDB server using mapping xml files.
Cisco	CSS	6.10, 7.4	SNMP	Mapping of Virtual IPs to real IP addresses of servers configured for load balancing; configuration files, load balancing algorithms, and end user IP addresses. **Note**: Cisco WebNS is the software version running on the 11000 and 11500 series CSS.

Vendor	Product	Versions	Credentials	Discovers...
Citrix	XEN	3.4	SSH, Telnet	Bridge, CPU, Execution Environment, File System, File System Export, Interface, Layer2Connection, Node, Physical Port, Virtualization Layer Software, Xen domain config.
EMC	EMC Control Center (ECC)	6.0.1	Oracle DB	Synchronized Configuration Items (CIs) currently include Storage Arrays, Fibre Channel Switches, Hosts (Servers), Storage Fabrics, Storage Zones, Logical Volumes, Host Bus Adapters, Storage Controllers, and Fibre Channel Ports. Integration also synchronizes physical relationships between various hardware and logical relationships between Logical Volumes, Storage Zones, Storage Fabrics, and hardware devices to enable end-to-end mapping of the storage infrastructure in UCMDB. **Note**: Synchronized content is discovered, not the application topology.
F5	BIG-IP LTM	4.6, 9.1	SNMP	Mapping of Virtual IPs to real IP addresses of servers configured for load balancing; configuration files, load balancing algorithms, and end user IP addresses.
HP	Network Node Manager (NNM)	8.1, 8.11, 9.0, 9.1	NNM API	Discovered nodes, IPs, networks, interfaces and Layer 2 connection information to create a Layer 2 topology in UCMDB.
HP	NonStop	H06.x	SSH	Database, Database Instance, HP NonStop, NonStop SQL/MX.
HP	nPartitions	A.03xx, A.04xx, A.05xx	SSH, Telnet	CPU, Fibre Channel HBA, File System, HP Complex, HP nPar Config, HP vPar Config, I/O Chassis, CellBoard, Interface, nodes, Physical Volume, SCSI Adapter, Volume Group
HP	ServiceGuard	11.1x	Shell	SG cluster software, SG packages, SG resources, cluster members

Vendor	Product	Versions	Credentials	Discovers...
HP	SIM	5.1, 5.2, 5.3, 6.0, 6.1, 6.2, 6.3	HP SIM	Synchronized configuration items (CIs) include nodes such as Windows, and UNIX servers, network devices, printers, clusters, cellular/partitioned systems, blade enclosures, and racks. Some server components, for example, CPU, are also synchronized. The integration also synchronizes relationships between blade servers and blade enclosures, virtual machines, physical servers, and so on. **Note**: Synchronized Content, not discovery of application topology.
HP	Storage Essentials (SE)	6.0.0; 6.3 9.4, 9.41, 9.5	SQL	Synchronized Configuration Items (CIs) including Storage Arrays, Fibre Channel Switches, Hosts (Servers), Storage Fabrics, Storage Zones, Logical Volumes, Host Bus Adapters, Storage Controllers, and Fibre Channel Ports. The integration also synchronizes physical relationships between various hardware and logical relationships between Logical Volumes, Storage Zones, Storage Fabrics, and hardware devices to enable end-to-end mapping of the storage infrastructure in UCMDB.
IBM	AS/400	V4R2M0, V3R2M1, V3R2M0, V4R5M0, V5R3, V6R1	AS400	AS400Agent, Interface, IpSubnet, Node.

Vendor	Product	Versions	Credentials	Discovers...
IBM	DB2 Universal Database (UDB)	8.2, 9.1, 9.5, 9.7	SQL	DB2 databases, including instances, tablespaces, users, processes, jobs (backup routines, log routines, and so on), any database objects. Discovery through: • direct connection to DB2 database, • SQL queries • HP DFM z/OS Mainframe **Note**: Discovery Agent, 9.2, 9.5 are recent versions.
IBM	HACMP	5.4	SSH, Telnet	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).
IBM	HMC	3.x, 5.x, 6.x, 7.x	SSH, Telnet	CPU, I/O Slot, IBM Frame, IBM HMC, IBM LPar Profile, IBM Processor Pool, Interface, Node, Virtualization Layer Software, SCSI Adapter, Physical Port, Physical Volume, Fibre Channel HBA, File System, SEA Adapter.
IBM	HTTP Server	5, 6.1, 7	Shell	IBM Http Server's WebSphere plug-in configuration by parsing the IHS plug-in configuration file.

Vendor	Product	Versions	Credentials	Discovers...
IBM	MQ Series (aka WebSphere MQ)	5.31, 6, 7.1	Shell	MQ subsystems at the system configuration level; DFM does not monitor or discover which active jobs or applications are running through the queues. Discovery includes Queue Managers, System Parameters, Queue-Sharing Groups, related DB2 Data-Sharing Groups, Cross Coupling Facility groups/members, Channel Initiator, Sender Channel, Server Channel, Receiver Channel, Requester Channel, Client Connection Channel, Server Connection Channel, Cluster Sender Channel, Cluster Receiver Channel, Alias Queue, Model Queue, Local Queue, Transmission Queue, Remote Queue, MQ Process, and MQ Cluster.
IBM	Websphere Application Server	5.x, 6.1, 7.0	Shell	J2EE Server, J2EE application, JDBC datasource, Database, EJB Module, Web Module, J2EE Domain and JMS resources
JBoss	Application Server	3.x, 4.x , 5.x	JMX	JBoss J2EE application server, EJB Module, Entity Bean, J2EE Application, J2EE Domain, JDBC Data Source, JMS Destination, JMS Server, JVM, Message Driven Bean, Servlet, Session Bean, Web module.
JBoss	Application Server	3.x, 4.x, 5.x	Shell	JBoss J2EE application server, EJB Module, Entity Bean, J2EE Application, J2EE Domain, JDBC Data Source, JMS Destination, JMS Server, JVM, Message Driven Bean, Servlet, Session Bean, Web module.
Microsoft	Active Directory	2000, 2003, 2008	LDAP	Forest, Sites, Sitelinks, Domain controllers, Networks, and so on.

Vendor	Product	Versions	Credentials	Discovers...
Microsoft	Cluster Services	Windows Server 2003, Windows Server 2008	Shell	Cluster software, configuration files, cluster members, MCS Resource Groups, MCS Resources.
Microsoft	Exchange Server	2003	WMI	Administrative Group, Directory Service Access DC, Exchange Folder, Exchange Folder Tree, Exchange Links, Exchange Message Queue, Exchange System, Routing Group.
Microsoft	Exchange Server	2003, 2007, 2010	LDAP	Forest, Sites, Exchange folders, folder trees, Administrative groups, Connectors.
Microsoft	Exchange Server	2007, 2010	NTCMD, PowerShell	Exchange Server, Exchange roles, Administrative group, Exchange Organization, Exchange Clustered Mailbox, Exchange Database Availability Group.
Microsoft	Hyper-V	Windows 2008, Windows 2008 R2	NTCMD, WMI	Resource pools, virtual switches, virtual NICs, virtual machines, and configuration files.
Microsoft	IIS	5, 6, 7	Shell	Discover the IIS Web Server, IIS Web Site, IIS virtual Dir, IIS Application pool, web services and configuration files.
Microsoft	Message Queue	3.0, 4.0, 5.2	LDAP, NTCMD	MSMQ Manager, MSMQ Routing Link, MSMQ Manager, MSMQ Queue, MSMQ Rule, MSMQ Trigger.
Microsoft	Network Load Balancer	2003, 2008	NTCMD	NLB Cluster, NLB Cluster Software and Node.
Microsoft	SharePoint	2007, 2010	NTCMD	Windows, SQL Server, IIS Application Pool, IIS Web Server, IIS Web Service, IIS Web Site, SharePoint Farm.

Vendor	Product	Versions	Credentials	Discovers...
Microsoft	SQL Server	7, 2000, 2005, 2008	SQL	Discovery of MS SQL databases, including instances, tablespaces, users, processes, jobs (backup routines, log routines, and so on), any database objects, MS SQL clustering, and log file shipping tasks.
NetApp	Data ONTAP	7.2.x, 7.3.x	NetApp	Node, LogicalVolume, Logical Volume Snapshot, FileSystem, FileSystemExport, IpAddress, Interface, CPU, Memory.
Nortel	Alteon	2424, 2208	SNMP	Mapping of Virtual IPs to real IP addresses of servers configured for load balancing; configuration files, load balancing algorithms, and end user IP addresses.
Oracle	Application Server	10g	Shell	OC4J groups, OC4J instances and its URLs.
Oracle	Database	9,10g,11g	Shell	Oracle database, TNS Listener software.
Oracle	Database	8, 9, 10g, 11g	SQL	Oracle databases, including SIDs, TNS names, instances, tablespaces, users, processes, jobs (backup routines, ONP, jobs, log routines, and so on), and any database objects.
Oracle	LDOM	1.0-1.3	SSH, Telnet	LDOM Networking and Storage topologies.
Oracle	Oracle VM Server for SPARC	2.0-2.1	SSH, Telnet	LDOM Networking and Storage topologies.
Oracle	RAC	9,10g,11g	Shell	Oracle RAC.
Oracle	RAC	10g	SQL	Oracle RAC.
Oracle	E-Business Suite	11i, 12	SQL	Oracle E-Business applications, such as Oracle Financials; infrastructure components, Web servers, application servers, individual components, and configuration files.

Vendor	Product	Versions	Credentials	Discovers...
Oracle	MySQL Database	3.x, 4.x, 5.0, 5.1, 6.0	Shell	Support MySQL Master-Master and Master-Slave configuration. Discover MySQL Database, configuration files, Replication job
Oracle	Siebel CRM	7.5, 7.7, 8.0, 8.1	Shell	Discovery of Siebel Enterprise, including Siebel applications (CallCenter, Financial, and so on), Siebel infrastructure components, Siebel Web servers, application servers, gateway servers, individual Siebel, components and configuration files.
Oracle	WebLogic	8.x, 9.x, 10.x, 11g, 11gR1 PS1, 11gR1 PS2	Shell or JMX	Weblogic J2EE Server, J2EE application, JDBC datasource, Database, EJB Module, Web Module and JMS resources, J2EE Domain, J2EE Cluster.
SAP	NetWeaver	2.x, 4, 7	JMX; SAP JCo	SAP ABAP Application Server, SAP Clients, SAP Gateway, SAP System, SAP Work Process, JDBC Data Sources, Databases, Hosts in deployment with IPs, SAP J2EE Application Server, SAP J2EE Dispatcher, SAP J2EE Server Process, SAP J2EE Central Services, J2EE domain, EJBs, EJB Modules, Entity Beans, Stateful/Stateless Session Beans, Web Module, SAP Business Process, SAP Business Scenario, SAP Process Step, SAP Project, SAP Transaction, SAP Application Components, SAP Transports, SAP ITS AGate, SAP ITS WGate.
SAP	SAP Solution Manager	6.4, 7.0	SAP JCo	SAP ABAP Application Server, SAP Clients, SAP System, JDBC Data Sources, Databases, SAP J2EE Application Server, SAP J2EE Dispatcher, SAP J2EE Central Services, J2EE domain.
SAP	SMD Agent	7.00-7.30	SSH, Telnet, NTCMD	SapSmdAgent, SAP Sytem

Vendor	Product	Versions	Credentials	Discovers...
SAP	TREX/BIA	7.00-7.30	SSH, Telnet, NTCMD	SapTrexInstance, SapTrexSystem, SAP System
SAP	Web Dispatcher	6.40, 7.00-7.30	SSH, Telnet, NTCMD	SapWebDispatcher, SAP System
Sun	MySQL Database Server	4.x and above	Shell	MySQL databases and MySQL replication topology.
Sun	Solaris Cluster	3.2	SSH, Telnet	Cluster Software, Configuration file, Execution Environment, Node, Sun Cluster, Sun Cluster Resource, Sun Resource Group.
Sun	Solaris Zones	5.1	Shell	Containers, zones, and share resources.
Sybase	Adaptive Server Enterprise	10.x, 11.x, 12.x, 15.0, 15.5	SQL	Sybase databases, including instances, tablespaces, users, processes, jobs (backup routines, log routines, and so on), and any database objects.
Symantec	Veritas Cluster Server (VCS) for UNIX	2.x, 3.x, 4.x, 5.x	Shell	Cluster Software, configuration files, cluster members, VCS Resource Groups, VCS Resources.
TIBCO	ActiveMatrix BusinessWorks	5.7, 5.8	SSH, Telnet, TIBCO	TibcoAdapter, TibcoAdministrationDomain, TibcoApplication, TibcoBusinessWorks, TibcoEmsServer, JMS Destination, JMS Server
TIBCO	Enterprise Message Server	6.0	SSH, Telnet, TIBCO	TibcoEmsServer, JMS Destination, JMS Server
Tomcat	Apache	5.x, 6.x	Shell	Tomcat Server instances, Web applications, configuration files, virtual servers, listening ports.
Troux	Troux	9.0x		
VMware	ESX	2.5, 3, 4, 4.1	Shell	

Vendor	Product	Versions	Credentials	Discovers...
VMware	ESX & ESXi	2.5, 3, 3i, 3.5, 4, 4.1	VIM	ESX servers, cluster groups, virtual resource groups.
VMware	vCenter (formerly Virtual Center)	2.01, 2.5, 4, 4.1	VIM and WMI	Virtual Center Server, License Server, ESX servers, cluster groups, virtual resource groups.
VMware	vCloud Director	1.5	vCloud	VMware vCloud Director and vCloud Resources (Organization, Catalog, Media, vApp, and so on).

Discovered Operating Systems

Vendor	Product	Versions	Credentials	Content
IBM	AIX	5.x, 6.x	SSH, Telnet	OS, Memory, Disks, CPU, Processes, Software (packages), Services (daemons), Files, Local Users
HP	HP-UX	10.xx, 11.xx	SSH, Telnet	OS, Memory, Disks, CPU, Processes, Software (packages), Services (Daemons), Files, Local Users, HP-UX Clusters
IBM	OS/390		SNMP	Simple mainframe discovery identifies Sysplex, LPARs, and IPs
IBM	z/OS	1.8, 1.9, 1.10, 1.11, 1.12	EView	CPU, Dasd3390, InstalledSoftware, Interface, IpAddress, IpServiceEndpoint, Mainframe CPC, MainframeMajorNode, MainframePageDataset, MainframeSubsystem, MainframeSysplex, MainframeXcfGroup, MainframeXcfMember, Node, Volume Group, zOS
RedHat	RedHat Enterprise Linux	3, 4, 5, 5.1, 5.2, 5.3, 5.4, 5.5	SSH, Telnet	OS, Memory, Disks, CPU, Processes, Software (packages), Services (daemons), Files, Local Users
Sun	Solaris	5.9, 5.10	SSH, Telnet	OS, Memory, Disks, CPU, Processes, Software (packages), Services (daemons), Files, Local Users
Microsoft	Windows	All Versions later than Windows 2000	NTCMD, PowerShell, WMI	OS, Memory, Disks, CPU, Processes, Software, Services, Files, Local Users

Supported Agents

The following agents are supported:

Agent	Description
SNMP Agent	Provides information about the operating systems, device types, installed software, and other system resources information. SNMP agents can usually be extended to support new MIBs, exposing more data for management purposes.
WMI Agent	Microsoft's remote management agent, which is usually available for access by a remote administrator. The WMI agent is also extensible by adding WMI providers to the generic agent.
Telnet/SSH Agent (or daemon)	Used mostly on UNIX systems to connect remotely to a machine and to launch various commands to obtain data.
Universal Discovery Agent	A remote administration technology similar in functionality to Telnet/SSH that enables launching any console command on Windows/UNIX/Mac OS X machines. The Universal Discovery Agent (UD Agent) implements a Web Services interface that is secured by SSL to provide private and encrypted communication between the Data Flow Probe and the UD Agent.
xCmd	A remote administration technology similar in functionality to Telnet/SSH that enables launching any console command on Windows machines. xCmd relies on Administrative Shares & Remove Service Administration APIs to function correctly. The **xCmd.exe** file is signed by an HP digital certificate. To validate that **xCmd.exe** is provided by HP, right-click the **xCmd.exe** file (or **xCmdSvc.exe** on a remote machine), select **Properties** and view the digital signatures.
Application specific	Depends on the remote application to function as an agent and respond appropriately to the Probe's remote queries, for example, database discoveries, Web server discoveries, and SAP and Siebel discoveries.

Universal Discovery Agent, Software Utilization Plug-In, Scanner and Software Library Support

The following table lists operating systems and details Universal Discovery Agent, Software Utilization Plug-In, Scanner and Software Library support for them.

Operating System	Platform	UD Agent	Utilization Plug-in	Scanner[1]	Software Library
Microsoft Windows XP Home	x86	x	x	x	x
Microsoft Windows XP Professional	x86	x	x	x	x
Microsoft Windows XP Professional	x64	x	x	x[2]	
Microsoft Windows XP Professional	ia64			x[2]	
Microsoft Windows Server 2003	x86	x	x	x	x
Microsoft Windows Server 2003 R2	x86	x	x	x	x
Microsoft Windows Server 2003	ia64			x[2]	
Microsoft Windows Server 2003	x64	x	x	x[2]	x
Microsoft Windows Server 2003 R2	x64	x	x	x[2]	x
Microsoft Windows Server 2008	x86	x	x	x	x
Microsoft Windows Server 2008	ia64			x[2]	
Microsoft Windows Server 2008	x64	x	x	x[2]	x
Microsoft Windows Server 2008 R2	x64	x	x	x[2]	x
Microsoft Windows Vista Business/Enterprise/Ultimate	x86	x	x	x	x
Microsoft Windows Vista Business/Enterprise/Ultimate	x64	x	x	x[2]	x
Microsoft Windows 7 Professional/Enterprise/Ultimate	x86	x	x	x	x
Microsoft Windows 7 Professional/Enterprise/Ultimate	x64	x	x	x[2]	x
Red Hat Enterprise Linux AS/ES/WS 3	x86, x64	x	x	x[3]	x
Red Hat Enterprise Linux AS/ES/WS 4	x86, x64	x	x	x[3]	x
Red Hat Enterprise Linux 5 Server/Desktop	x86, x64	x	x	x[3]	x

Operating System	Platform	UD Agent	Utilization Plug-in	Scanner[1]	Software Library
Red Hat Enterprise Linux 6 Server/Workstation	x86, x64	x	x	x[3]	x
Novell SUSE Linux Enterprise Server/Desktop 9	x86, x64	x	x	x[3]	x
Novell SUSE Linux Enterprise Server/Desktop 10	x86, x64	x	x	x[3]	x
Novell SUSE Linux Enterprise Server/Desktop 11	x86, x64	x	x	x[3]	x
Oracle Linux 4	x86, x64	x	x	x[3]	x
Oracle Linux 5	x86, x64	x	x	x[3]	x
Oracle Linux 6	x86, x64	x	x	x[3]	x
CentOS 5	x86, x64	x	x	x[3]	x
CentOS 6	x86, x64	x	x	x[3]	x
Ubuntu Linux Server/Desktop 10	x86, x64	x	x	x[3]	
Ubuntu Linux Server/Desktop 11	x86, x64	x	x	x[3]	
IBM AIX 5L 5.3	POWER	x	x	x	x
IBM AIX 6.1	POWER	x	x	x	x
IBM AIX 7.1	POWER	x	x	x	x
Oracle Solaris 9	SPARC	x	x	x	x
Oracle Solaris 10	SPARC	x	x	x	x
Oracle Solaris 10	x86, x64	x	x	x	x
Oracle Solaris 11	SPARC	x	x	x	x
Oracle Solaris 11	x86, x64	x	x	x	x
HP HP-UX 11.11 (11i)[4]	HPPA	x	x	x	x
HP HP-UX 11.23 (11i v2)[4]	HPPA	x	x	x	x
HP HP-UX 11.23 (11i v2)	ia64	x	x	x	x
HP HP-UX 11.31 (11i v3)[4]	HPPA	x	x	x	x
HP HP-UX 11.31 (11i v3)	ia64	x	x	x	x
Apple Mac OS X 10.4	x86	x	x	x	x

Operating System	Platform	UD Agent	Utilization Plug-in	Scanner[1]	Software Library
Apple Mac OS X 10.5	x86	x	x	x	x
Apple Mac OS X 10.6	x86	x	x	x	x
Apple Mac OS X 10.7	x86	x	x	x	x

Notes

1. Unless otherwise noted, scanners are natively supported.
2. Windows 64-bit architectures are supported by the 32-bit Windows (x86) scanner.
3. Linux 64-bit architectures are supported by the 32-bit Linux (x86) scanner.
4. HPPA is PA-RISC 2.0 architecture.

For details on data collected by scanners, see the *Data Collected by the Scanners Guide*.

Supported Protocols

This section describes the credentials for the supported protocols for the Discovery and Integration Content Pack. For information about setting up protocol credentials in UCMDB, see the section about setting up the Data Flow Probe in the *HP Universal CMDB Data Flow Management Guide*.

Note: Credential attributes must not contain non-English letters.

AS400 Protocol

Parameter	Description
User	The user used on the AS400 system to execute the discovery commands.
Password	The password for the user account on the AS400 system used to execute the discovery commands.

AWS Protocol

Parameter	Description
User Name	Access Key ID. An alphanumeric text string that uniquely identifies the owner of the account.
User Password	Secret Access Key, performing the role of a password.

CA CMDB Protocol

Parameter	Description
User Name	The username used by CA CMDB's GRLoader to connect to CA CMDB remotely.
User Password	The password used by CA CMDB's GRLoader to connect to CA CMDB remotely.

Generic DB Protocol (SQL)

Parameter	Description
Database Type	The database type. Select the appropriate type from the box. The following database types are supported: • DB2 • Microsoft SQL Server • Microsoft SQL Server (NTLM) • Microsoft SQL Server (NTLM v2) • MySQL • Oracle • Sybase
Port Number	The port number on which the database server listens. • If you enter a port number, DFM tries to connect to a SQL database using this port number. • **For an Oracle database**: If there are many Oracle databases in the environment and you do not want to have to create a new credential for each separate database port, you leave the Port Number field empty. When accessing an Oracle database, DFM refers to the **portNumberToPortName.xml** file and retrieves the correct port number for each specific Oracle database port. **Note**: You can leave the port number empty on condition that: • All Oracle database instances are added to the **portNumberToPortName.xml** file. For details, see "portNumberToPortName.xml File" on page 64. • The same user name and password is needed to access all Oracle database instances.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the database.
User Name	The name of the user needed to connect to the database.
Password	The password of the user needed to connect to the database.
Instance Name	The name of the database instance, that is, the Oracle system identification or the DB2 database name. When connecting to any database, you can leave this field empty. In this case, DFM takes the SID from the Triggered CI data value: **${DB.name:NA}**.

Parameter	Description
Encryption method	• **None**. • **SSL**. For Oracle only.
Trust Store File Path	Enter the full path to the SSL trust store file. To use the trust store file, do one of the following: • Enter the name (including the extension) and place the file in the following resources folder: **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources** • Insert the trust store file full path.
Trust Store Password	The SSL trust store password.

Generic Protocol

This protocol is intended for integrations that do not need a specific protocol. It is recommended to use this protocol for all out-of-the-box integrations, as they require a user name and password only.

Parameter	Description
Description	Description of the credentials.
User Name	The name of the user needed for authentication.
User Password	The password of the user needed for authentication.

HP Asset Manager Protocol

Parameter	Description
Asset Manager User Name	The name of the AM user.
Asset Manager Password	The password of the AM user.
DB User Name	The name of the AM database user.
DB Password	The password of the AM database user.

HP SIM Protocol

Parameter	Description
Port Number	The port at which the SIM MXPartner WebService API listens for SOAP requests. The defaults are **280** for HTTP and **50001** for HTTPS.
SIM Database Instance	• **Microsoft SQL Server**: Enter the instance name only for non-default instances of Microsoft SQL Server. • **Oracle**: Enter the SID.
SIM Database Name	(Microsoft SQL Server only) Enter the name of the database.
SIM Database Password	The password of the database user (Microsoft SQL Server) or schema name (Oracle) for the SIM database.
SIM Database Port	The listener port for the database.
SIM Database Type	The SIM Database type: • MSSQL • MSSQL_NTLM • Oracle
SIM Database User Name	The database user (Microsoft SQL Server) or schema name (Oracle) with permissions to access the database.
SIM Webservice Protocol	Choose between **HTTP** or **HTTPS**.
User Name	The name of the user needed to connect to the application.
User Password	The password of the user needed to connect to the application.

JBoss Protocol

Parameter	Description
Port Number	The port number.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the JBoss application server.
User Name	The name of the user needed to connect to the application.
Password	The password of the user needed to connect to the application.

LDAP Protocol

Parameter	Description
Port Number	The port number.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the LDAP application server.
User Name	The name of the user needed to connect to the application.
Password	The password of the user needed to connect to the application.
Protocol	Choose which security model to use to access the service: • **LDAP**. Discovery uses an unprotected connection. • **LDAPS**. Discovery uses an SSL connection.
LDAP Authentication Method	**Simple**. The supported authentication method.
Trust Store File Path	The file containing trusted certificates. To import certificates into the Trust Store file: • Create a new Trust Store or use the default Java Trust Store: `<java-home>/lib/security/cacerts` • Enter the full path to the LDAP Trust Store file.
Trust Store Password	The LDAP Trust Store password used to access the Trust Store file. This password is set during the creation of a new Trust Store. If the password has not been changed from the default, use **changeit** to access the default Java Trust Store.

NetApp Protocol

Parameter	Description
NetApp ONTAPI Protocol	The protocol type. **Default:** https
Port Number	The port number. **Default:** 443
User Name	The name of the user needed to connect to the application.
User Password	The password of the user needed to connect to the application.

NNM Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Data Flow Probe stops trying to connect to the NNMi server.
NNM Password	The password for the specified NNMi Web service (for example, `Openview`).
NNM User name	The user name for connecting to the NNMi console. This user must have the NNMi Administrator or Web Service Client role.
NNM Webservice Port	The port for connecting to the NNMi console. This field is pre-filled with the port that the JBoss application server uses for communicating with the NNMi console, as specified in the following file: • **Windows:** `%NnmDataDir%\shared\nnm\conf\nnm.ports.properties` • **UNIX:** `$NnmDataDir/shared/nnm/conf/nnm.ports.properties` For non-SSL connections, use the value of jboss.http.port, which is `80` or `8004` by default (depending on the presence of another Web server when NNMi was installed). For SSL connections, use the value of **jboss.https.port**, which is `443` by default.
NNM Webservice Protocol	The protocol for the NNMi Web service (the default is **http**).
UMCBD Password	The password for the UCMDB Web service (the default is **admin**).
UCMDB Username	A valid UCMDB Web service account name with the UCMDB Administrator role (the default is **admin**).
UCMDB Webservice Port	The port for connecting to the UCMDB Web service. If you are using the default UCMDB configuration, use port **8080** (for non-SSL connections to UCMDB).
UCMDB Webservice Protocol	The protocol for the UCMDB Web service (the default is **http**).

NTCMD Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the NTCMD server.
User Name	The name of the user needed to connect to the host as administrator.
Password	The password of the user needed to connect to the host as administrator.
Windows Domain	The Windows domain in which the credentials are defined. If this field is left empty or is not a valid domain, the NTCMD protocol assumes the user is defined locally on the host.
Run remote commands impersonated	If selected, the commands for the discovery are executed remotely under the **User Name** of this credential. If not selected, the commands for the discovery are, instead, executed remotely under the **LocalService** account.
Remote Share Path	Used where **Admin$** does not exist on the Windows machine being connected to. Type here the name of the SHARE concatenated with full path to the Windows directory of the machine being connected to. For example: **Share$\Windows**
Share Local Path	The full path to the Windows directory of the machine being connected to. For example: **C:\Windows**

Note: This protocol uses the DCOM protocol for connecting to remote machines. The DCOM protocol requires that the following ports are open: 135, 137, 138, and 139. In addition the DCOM protocol uses arbitrary ports between 1024 and 65535, but there are ways to restrict the port range used by WMI/DCOM/RPC. In addition, for information about for configuring DCOM to work with firewalls, see http://support.microsoft.com/kb/154596/en-us. For all versions of Windows after NT, port 445 (name: microsoft-ds) is the preferred port for resource sharing, including Windows file sharing and other services. It uses the TCP Protocol and replaces ports 137-139.

PowerShell Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the destination machine.
User Name	The name of the user that can connect to the remote machine by PowerShell.
User Password	The password of the user that can connect to the remote machine by PowerShell.
Windows Domain	The Windows domain on which the credentials are defined. If this field is empty, PowerShell assumes that the user is defined locally on the host.

Remedy Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Data Flow Probe stops trying to connect to the Remedy application server.
Remedy Password	Enter the password of the user account that enables access to Remedy/Atrium through the Java API.
Remedy Username	Enter the user name that enables access to Remedy/Atrium through the Java API.

SAP JMX Protocol

Parameter	Description
Port Number	The SAP JMX port number. The SAP JMX Port structure is usually `5<System Number>04`. For example, if the system number is `00`, the port is `50004`. Leave this field empty to try to connect to the discovered SAP JMX port; SAP JMX port numbers are defined in the **portNumberToPortName.xml** configuration file. For details, see "portNumberToPortName.xml File" on page 64.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the SAP JMX console.
User Name	The name of the user needed to connect to the application as administrator.
Password	The password of the user needed to connect to the application as administrator.

SAP Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the SAP console.
User Name	The name of the user needed to log in to the SAP system. The user should have the following permissions: **Authorization Object: S_RFC** Authorization: For the **S_RFC** object, obtain privileges: RFC1, SALX, SBDC, SDIF, SDIFRUNTIME, SDTX, SLST, SRFC, STUB, STUD, SUTL, SXMB, SXMI, SYST, SYSU, SEU_COMPONENT. **Authorization Object: S_XMI_PROD** Authorization: EXTCOMPANY=MERCURY; EXTPRODUCT=DARM; INTERFACE=XAL **Authorization Object:S_TABU_DIS** Authorization: DICBERCLS=SS; DICBERCLS=SC
Password	The password of the user needed to log in to the SAP system.
SAP Client Number	It is recommended to use the default value (**800**).
SAP Instance Number	By default, set to **00**.
SAP Router String	A route string describes the connection required between two hosts using one or more SAProuter programs. Each of these SAProuter programs checks its Route Permission Table (http://help.sap.com/saphelp_nw04/helpdata /en/4f/992dfe446d11d189700000e8322d00/content.htm) to see whether the connection between its predecessor and successor is allowed. If it is, SAProuter sets it up.

Siebel Gateway Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the Siebel Gateway console.
User Name	The name of the user needed to log on to the Siebel enterprise.
Password	The password of the user needed to log on to the Siebel enterprise.
Siebel Site Name	The name of the Siebel Enterprise.
Path to Siebel Client	The location on the Probe machine of the Siebel driver folder, where you copied `srvrmgr`. For details, see the prerequisites section of the discovery task in "Siebel Discovery" on page 240. • If there are several protocol entries with different `srvrmgr` versions, the entry with the newer version should appear before the entry with the older version. For example, to discover Siebel 7.5.3. and Siebel 7.7, define the protocol parameters for Siebel 7.7 and then the protocol parameters for Siebel 7.5.3. • **Siebel discovery**. 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. These drivers usually exist on a 32-bit machine and can be copied to the 64-bit machine.
Port number	The port to use during connection to the Siebel Gateway. **Default:** empty.

SNMP Protocol

Parameter	Description
Port Number	(For SNMP versions v1, v2, and v3) The port number on which the SNMP agent listens.
Connection Timeout	Timeout(in milliseconds) after which the Probe stops trying to connect to the SNMP agent.
Retry Count	The number of times the Probe tries to connect to the SNMP agent. If the number is exceeded, the Probe stops attempting to make the connection.
Versions 1, 2	**Community**. Enter the authentication password you used when connecting to the SNMP service community (which you defined when configuring the SNMP service—for example, a community for read-only or read/write).

Parameter	Description
Version 3	**Authentication Method**: Select one of the following options for securing the access to management information: • **noAuthNoPriv.** Using this option provides no security, confidentiality, or privacy at all. It can be useful for certain applications, such as development and debugging, to turn security off. This option requires only a user name for authentication (similar to requirements for v1 and v2). • **authNoPriv.** The user logging on to the management application is authenticated by the SNMP v3 entity before the entity allows the user to access any of the values in the MIB objects on the agent. Using this option requires a user name, password, and the authentication algorithm (HMAC-MD5 or HMAC-SHA algorithms). • **authPriv.** The user logging on to the management application is authenticated by the SNMP v3 entity before the entity allows the user to access any of the values in the MIB objects on the agent. In addition, all of the requests and responses from the management application to the SNMP v3 entity are encrypted, so that all the data is completely secure. This option requires a user name, password, and an authentication algorithm (HMAC-MD5 or HMAC-SHA). **User Name**: The name of the user authorized to log on to the management application. **Password**: The password used to log on to the management application. **Authentication Algorithm**: The MD5 and SHA algorithms are supported. **Privacy Key**: The secret key used to encrypt the scoped PDU portion in an SNMP v3 message. **Privacy Algorithm**: The DES, 3DES, AES-128, AES-192 and AES-256 algorithms are supported.

Note: By default, SNMP queries are executed with a timeout of 3000 milliseconds. This value is defined in "snmpGlobalRequestTimeout" parameter in the globalSettings.xml configuration file.

Note: Due to control restrictions for some countries, the JDK has a deliberate, built-in key size restriction. If required (for example, if SNMP agents use 256-bit AES encryption), the restriction can be removed as follows:

1. Download the .zip file from http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html.

2. Extract **local_policy.jar** and **US_export_policy.jar** from the .zip file.
3. Copy these files and replace the files that arrived with the probe installation in the **${PROBE_INSTALL}\bin\jre\lib\security** folder.
4. Restart the probe.

Troubleshooting and Limitations

Problem. Failure to collect information from SNMP devices.

- **Solution 1:** Verify that you can actually access information from your Network Management station by using a utility that can verify the connectivity with the SNMP agent. An example of such a utility is **GetIf**.
- **Solution 2:**: Verify that the connection data to the SNMP protocol has been defined correctly.
- **Solution 3:** Verify that you have the necessary access rights to retrieve data from the MIB objects on the SNMP agent.

SSH Protocol

Note: If you use the SSH or Telnet credentials for discovery, we recommend that you add the following folders to the system path:

- /sbin
- /usr/sbin
- /usr/local/sbin

For details on configuring F-Secure when discovering Windows machines on which the F-Secure application is running on an SSH server, see "Host Connection by Shell Job" on page 910.

Parameter	Description
Port Number	By default an SSH agent uses port 22. If you are using a different port for SSH, enter that port number.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the remote machine. For the UNIX platform: If your server is slow, it is recommended to change Timeout to 40000.
Version	**SSH2**. Connect through SSH-2 only. **SSH1**. Connect through SSH-1 only. **SSH2 or SSH1**. Connect through SSH-2 and in case of error (if SSH-2 is not supported by the server), try to connect through SSH-1.

Parameter	Description
Shell Command Separator	The character that separates different commands in a shell (to enable the execution of several commands in the same line). For example, in UNIX, the default shell command separator is a semicolon (**;**). In Windows, the shell command separator is an ampersand (**&**).
Authentication Method	Choose one of the following authentication options to access SSH: • **password**. Enter a user name and password. • **publickey**. Enter the user name and path to the key file that authenticates the client. **Note:** The SSH1 protocol does not support public keys of the SSH2 protocol. Therefore, if you select **publickey** for the **Authentication Method**, you should **not** select **SSH1 or SSH2** for the **Version**. Instead, select the exact SSH version. • **keyboard-interactive**. Enter questions and answers. For details, see "Prompts and Responses" on next page below.
User Name	The name of the user needed to connect to the host through the SSH network protocol.
Password	The password of the user needed to connect to the host.
Key File Path	(Enabled when the `publickey` authentication method is selected.) Location of the authentication key. (In certain environments, the full key path is required to connect to an SSH agent.) **Note**: Enter the full path to the key file on the Probe machine.

Parameter	Description
Prompts and Responses	(Enabled when the `keyboard-interactive` authentication method is selected.) A method whereby the server sends one or more prompts to enter information and the client displays them and sends back responses keyed-in by the user. The following is an example of prompts and expected responses: **Prompt**: Please enter your user name. **Response**: Shelly-Ann **Prompt**: What is your age? **Response**: 21 **Prompt**: This computer is HP property. Press y to enter. **Response**: y To create these prompts and responses, enter the following strings in the fields, separated by commas: **Prompts**: user,age,enter **Response**: Shelly-Ann,21,y You can enter the full string as it appears in the SSH prompt, or you can enter a key word, for example, **user**. DFM maps this word to the correct prompt.
Sudo paths	The full paths to the **sudo** command. Paths are separated by commas.

Parameter	Description
Sudo commands	A list of commands that can be executed with the **sudo** command. Commands are separated by commas. For all commands to be executed with **sudo**, add an asterisk (*****) to this field. This field accepts a **sudo** command that prompts for the user's password. There is both pattern matching and pattern completion using Python/Jython regular expressions. For example,for the expressions: • **/usr/sbin/uname** • **uname -a** • **uname -r** • **/mypath/my_other_path/uname -my args -my other args** The pattern match would be: • **.*uname** This matches anything before **uname**, and any arguments **uname** has. **Note:** The list of commands that can be executed with **sudo** is dependant on the configuration of **sudo** commands on the discovered destination. Therefore, an asterisk (*) in this field means that all commands configured on the discovered destination should be run with **sudo**.
***SU username**	The name of the user to use with the **su** command.
***SU password**	The password to use for the **su** command.
Sudo/SU Policy	• **su.** Use the **su** command. • **sudo.** Use the **sudo** command. • **sudo or su.** Use the **sudo** command. In case of failure, use the **su** command.

*To configure SU support options, right click **SSH protocol** and select **Edit using previous interface**.

Note: The SSH1 protocol does not support public keys of the SSH2 protocol. Therefore, it is not advisable to set the alternative version (“SSH2 or SSH1”) if Authentication Method is configured to use publickey. In such a case, you should configure using the exact SSH protocol.

Troubleshooting

Problem. Failure to connect to the TTY (SSH/Telnet) agent.

- **Solution**. To troubleshoot connectivity problems with the TTY (SSH/Telnet) agent, use a utility that can verify the connectivity with the TTY (SSH/Telnet) agent. An example of such a utility is the client tool PuTTY.

Problem. Discovery job(s) fail with error message “Time out exception”.

- **Solution 1**. Increase the value of the **shellGlobalCommandTimeout** parameter in **globalSettings.xml**.
- **Solution 2**. Check the shell of the discovery user on the discovered destination. The command line for the ksh(korn shell) has a limit of 256 characters. Some discovery commands exceed that limit and can cause a “Time out exception” error message. In this case (a) Change the default shell for the discovery user from ksh to bash; or (b) Consult with the system administrator to determine if it is possible to increase the maximum command line size for korn shell on the problematic destination.

Telnet Protocol

Note: If you use the SSH or Telnet credentials for discovery, we recommend that you add the following folders to the system path:

- /sbin
- /usr/sbin
- /usr/local/sbin

Parameter	Description
Port Number	The port number. By default a Telnet agent uses port 23. If you are using a different port for Telnet in your environment, enter the required port number.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the remote machine. **For UNIX platforms**: If your server is slow, it is recommended to change Connection Timeout to 40000.
Authentication Method	Choose one of the following authentication options to access Telnet: • **password**. Enter a user name and password. • **keyboard-interactive**. Enter questions and answers. For details, see "Prompts and Responses" on next page below.
User Name	The name of the user needed to connect to the host.
Password	The password of the user needed to connect to the host.

Parameter	Description
Prompts and Responses	(Enabled when the `keyboard-interactive` authentication method is selected.) A method whereby the server sends one or more prompts to enter information and the client displays them and sends back responses keyed-in by the user. The following is an example of prompts and expected responses: **Prompt**: Please enter your user name. **Response**: Shelly-Ann **Prompt**: What is your age? **Response**: 21 **Prompt**: This computer is HP property. Press y to enter. **Response**: y To create these prompts and responses, enter the following strings in the fields, separated by commas: **Prompts**: user,age,enter **Response**: Shelly-Ann,21,y You can enter the full string as it appears in the Telnet prompt, or you can enter a key word, for example, **user**. DFM maps this word to the correct prompt.
Sudo paths	The full paths to the **sudo** command. Paths are separated by commas.

Parameter	Description
Sudo commands	A list of commands that can be executed with the **sudo** command. Commands are separated by commas. For all commands to be executed with **sudo**, add an asterisk (*****) to this field. This field accepts a **sudo** command that prompts for the user's password. There is both pattern matching and pattern completion using Python/Jython regular expressions. For example,for the expressions: • **/usr/sbin/uname** • **uname -a** • **uname -r** • **/mypath/my_other_path/uname -my args -my other args** The pattern match would be: • **.*uname** This matches anything before **uname**, and any arguments **uname** has. **Note:** The list of commands that can be executed with **sudo** is dependant on the configuration of **sudo** commands on the discovered destination. Therefore, an asterisk (*) in this field means that all commands configured on the discovered destination should be run with **sudo**.
***SU username**	The name of the user to use with the **su** command.
***SU password**	The password to use with the **su** command.
Sudo/SU policy	• **su.** Use the **su** command. • **sudo.** Use the **sudo** command. • **sudo or su.** Use the **sudo** command. In case of failure, use the **su** command. This is the default.

*To configure SU support options, right click **Telnet protocol** and select **Edit using previous interface**.

Troubleshooting and Limitations

Problem. Failure to connect to the TTY (SSH/Telnet) agent.

- **Solution**. To troubleshoot connectivity problems with the TTY (SSH/Telnet) agent, use a utility that can verify the connectivity with the TTY (SSH/Telnet) agent. An example of such a utility is the client tool PuTTY.

Limitation. The Telnet protocol does not support discovery of Windows Telnet servers.

Problem. Discovery job(s) fail with error message “Time out exception”.

- **Solution 1**. Increase the value of the **shellGlobalCommandTimeout** parameter in **globalSettings.xml**.
- **Solution 2**. Check the shell of the discovery user on the discovered destination. The command line for the ksh(korn shell) has a limit of 256 characters. Some discovery commands exceed that limit and can cause a "Time out exception" error message. In this case (a) Change the default shell for the discovery user from ksh to bash; or (b) Consult with the system administrator to determine if it is possible to increase the maximum command line size for korn shell on the problematic destination.

TIBCO Protocol

Parameter	Description
User Name	The name of the user needed to log into the TIBCO system.
Password	The password of the user needed to log into the TIBCO system.

UDDI Registry Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the UDDI Registry.
UDDI Registry URL	The URL where the UDDI Registry is located.

Universal Discovery Protocol

Parameter	Description
UD SHA1 ID	A hash of UD credential's certificates. Enables you to visually distinguish between UD credentials that have different certificates (different hash) and those that have similar certificates (similar hash). **Note:** This value is generated automatically and cannot be modified.
Port Number	The port number on which the UD Agent listens. Select one of the following ports: • **2738** • **7738**
Connection Timeout	The amount of time (in milliseconds) after which the Probe stops trying to connect to the UD Agent.

vCloud Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the vCloud application server.
User Name	The name of the user needed to connect to the application.
User Password	The password of the user needed to connect to the application.
vCloud Organization	The organization the user belongs to. When connecting with the global vCloud Administrator, set this to **System**.

VMware Infrastructure Management (VIM) Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to VMware Infrastructure.
Port Number	DFM uses the number defined here when processing one of the `Network - VMware` jobs: If the port number is left empty, DFM performs a WMI query to extract the port number from the registry. DFM queries **HKLM\SOFTWARE\VMware, Inc.\VMware VirtualCenter** and searches for the **HttpsProxyPort** or **HttpProxyPort** attributes: • If the **HttpsProxyPort** attribute is found, DFM uses its value for the port and sets the prefix to **HTTPS**. • If the **HttpProxyPort** attribute is found, DFM uses its value for the port and sets the prefix to **HTTP**.
Use SSL	**true**: DFM uses a Secure Sockets Layer (SSL) protocol to access VMware Infrastructure, and the prefix is set to **HTTPS**. **false**: DFM uses the http protocol.
User Name	The name of the user needed to connect to VMware Infrastructure.
Password	The password of the user needed to connect to VMware Infrastructure.

WebLogic Protocol

Parameter	Description
Port Number	If you enter a port number, DFM tries to connect to WebLogic using this port number. However, say you know that there are many WebLogic machines in the environment and do not want to have to create a new credential for each machine. You leave the Port Number field empty. When accessing a WebLogic machine, DFM refers to the WebLogic port (defined in **portNumberToPortName.xml**) already found on this machine (by TCP scanning). **Note**: You can leave the port number empty on condition that: • All WebLogic ports are added to the **portNumberToPortName.xml** file. For details, see "portNumberToPortName.xml File" on page 64. • The same user name and password is needed to access all WebLogic instances.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the WebLogic application server.
User Name	The name of the user needed to connect to the application.
Password	The password of the user needed to connect to the application.
Protocol	An application-level protocol that determines whether DFM should connect to the server securely. Enter **http** or **https**.
Trust Store File Path	Enter the full path to the SSL trust store file. To use the trust store file, do one of the following: • Enter the name (including the extension) and place the file in the following resources folder: **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\j2ee\weblogic\<WebLogic version>**. • Insert the trust store file full path.
Trust Store Password	The SSL trust store password.

Parameter	Description
Key Store File Path	Enter the full path to the SSL keystore file. To use the keystore file, do one of the following: • Enter the name (including the extension) and place the file in the following resources folder: **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\j2ee\weblogic\<WebLogic version>**. • Insert the keystore file full path.
Key Store Password	The password for the keystore file.

WebSphere Protocol

Parameter	Description
Port Number	The protocol port number as provided by the WebSphere system administrator. You can also retrieve the protocol port number by connecting to the Administrative Console using the user name and password provided by the WebSphere system administrator. In your browser, enter the following URL: **http:/<host>:9060/admin**, where: • **<host>** is the IP address of the host running the WebSphere protocol • **9060** is the port used to connect to the WebSphere console Access **Servers > Application Servers > Ports > SOAP_ CONNECTOR_ADDRESS** to retrieve the required port number.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the WebSphere server.
User Name	The name of the user needed to connect to the application.
Password	The password of the user needed to connect to the application.
Trust Store File Path	The name of the SSL trust store file. To use the trust store file, do one of the following: • Enter the name (including the extension) and place the file in the following resources folder: **C:\hp\UCMDB\DataFlowProbe\runtime\ probeManager\discoveryResources\j2ee\websphere**. • Insert the trust store file full path.
Trust Store Password	The SSL trust store password.
Key Store File Path	The name of the SSL keystore file. To use the keystore file, do one of the following: • Enter the name (including the extension) and place the file in the following resources folder: **C:\hp\UCMDB\DataFlowProbe\runtime\ probeManager\discoveryResources\j2ee\websphere**. • Insert the keystore file full path.
Key Store Password	The password for the keystore file.

WMI Protocol

Parameter	Description
User Name	The name of the user needed to connect to the host.
Password	The password of the user needed to connect to the host.
Windows Domain	The Windows domain in which the credentials are defined. If this field is left empty or is not a valid domain, the WMI protocol assumes the user is defined locally on the host.

Note: This protocol uses the DCOM protocol for connecting to remote machines. The DCOM protocol requires that the following ports are open: 135, 137, 138, and 139. In addition the DCOM protocol uses arbitrary ports between 1024 and 65535, but there are ways to restrict the port range used by WMI/DCOM/RPC. In addition, for information about for configuring DCOM to work with firewalls, see http://support.microsoft.com/kb/154596/en-us.

Default Ports for Supported Protocols

The following table lists the default ports for each supported protocol.

Protocol	Default Port
HP SIM	50001, 280
HTTP	80
JBoss	1099
LDAP	389
NNM	80
NTCMD	135, 137, 138, 139
PowerShell	80, 443, 5985, 5986 **Note:** The ports depend on the Microsoft Windows operating system configuration
SAP	• 3200 • 3300-3303 • 33xx, where xx is the SAP server instance number **Note:** To enable UCMDB to identify other port numbers mapped to SAP instances, you must configure the **portNumberToPortName.xml** file. For more details, see "How to Define a New Port" on page 46.
SAP JMX	• 50004, 50104, 50204, 50304, 50404 • 5xx04, where xx is the SAP J2EE server instance number **Note:** To enable UCMDB to identify other port numbers mapped to SAP instances, you must configure the **portNumberToPortName.xml** file. For more details, see "How to Define a New Port" on page 46.
Siebel Gateway	2320
SNMP	161
SQL	1521, 1433, 6789, 3306, 2048
SSH	22
Telnet	23
UDDI	80, 443
VMWare VIM	80, 443

Protocol	Default Port
WebLogic	7001, 7002
WebSphere	8880
WMI	135, 137, 138, 139

Supported Integrations

- Aperture VISTA
- Atrium to UCMDB
- CA CMDB
- CiscoWorks LMS
- Data Dependency and Mapping Inventory
- Data Push into Atrium
- EMC Control Center (ECC)
- HP Asset Manager
- HP Configuration Manager
- HP ServiceCenter/Service Manager
- HP Systems Insight Manager (HP SIM)
- IDS Scheer ARIS
- Microsoft SCCM/SMS
- NetApp SANscreen/OnCommand Insight
- Network Node Manager (NNMi)
- ServiceNow
- Storage Essentials (SE)
- Troux
- UCMDB to XML Adapter

Support for HP UCMDB Integration Service on Linux

The following table lists the integration adapters that support the HP UCMDB Integration Service on the Linux platform.

Adapter	Population	Federation	Data push
AM	Not supported	Not supported	Not supported
SM 6.2x\7.0x\7.1x-9.2x	-	Not supported	Not supported
SM 9.x	Supported	Supported	Supported
UCMDB 9.x\10.x	Supported	Supported	-
CM policy\kpi adapters	-	Supported	-

Adapter	Population	Federation	Data push
DDMi	Not supported	Supported	-
Generic Push adapters	-	-	Not supported
SMS	Not supported	Supported	-
Service now	-	-	Not supported
Jython based integration adapters	Not supported	-	-

Localization

This section details localized versions of operating systems and applications which are supported by UCMDB.

Operating Systems

Discovery of host resources, Universal Discovery Agent installation (including the Software Utilization Plug-In) and inventory discovery using Inventory Scanners, is supported for the following localized versions of **Windows**:

- Chinese
- Dutch
- French
- German
- Italian
- Japanese
- Korean
- Portuguese
- Russian
- Spanish

Applications

Vendor	Product	Versions	Supported Localized Versions
Microsoft	Active Directory	2003, 2008	Japanese
Microsoft	Cluster Services	2003R2, 2008R2	Japanese
Microsoft	Hyper-V	2008, 2008R2	Japanese

Part II: Applications

Chapter 3

Active Directory Discovery

This chapter includes:

Overview

Active Directory (AD) provides an extensible and scalable directory service that enables efficient managing of network resources.

DFM discovers Active Directory topology through the LDAP Directory Service Interface that communicates with the AD domain controllers. DFM uses JNDI to provide the API that interacts with the LDAP Directory Service Interface.

Supported Versions

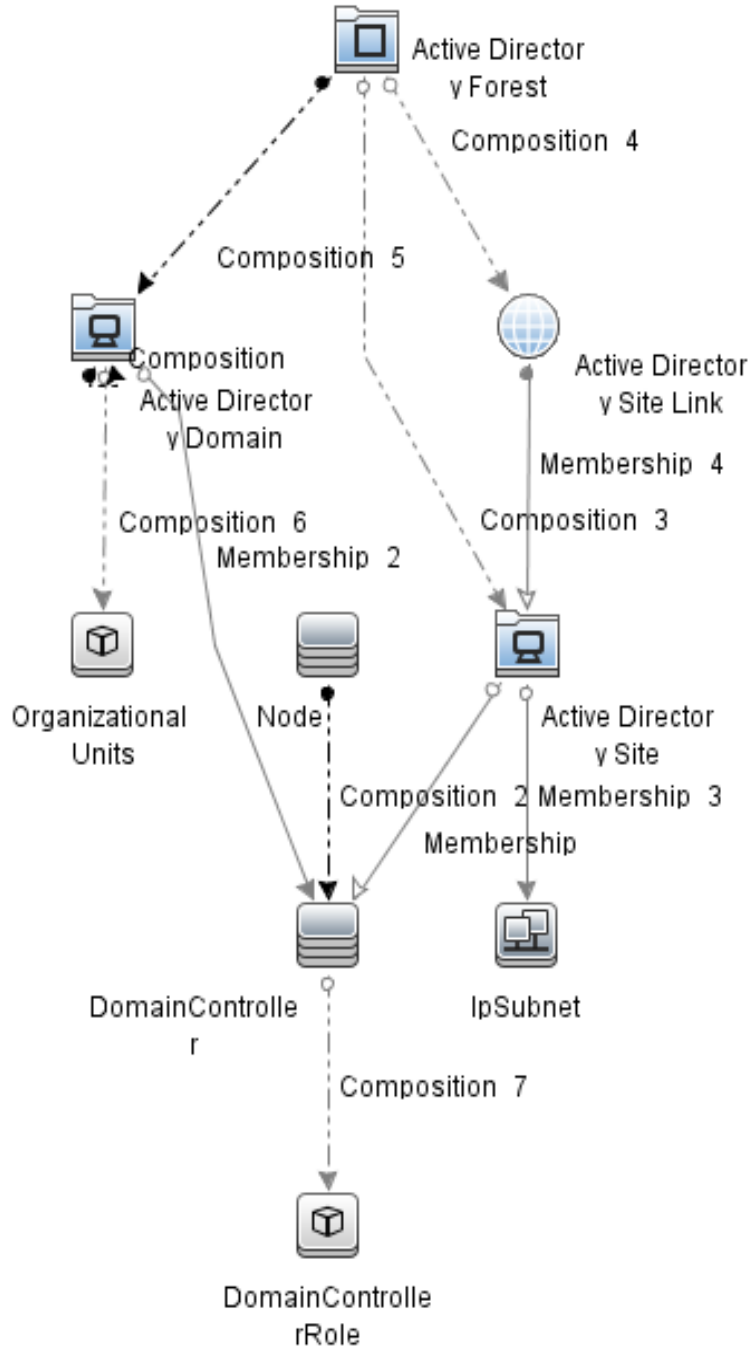
This discovery solution supports the following servers:

- 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 123.



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**. Version 2 has never been standardized in any formal specification. Therefore, DFM uses the version 3 protocol.

      Note: User Name: if a domain is present, use **username@domain**.

   For credential information, see "Supported Protocols" on page 82.

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 "portNumberToPortName.xml File" on page 64 and "How to Define a New Port" on page 46.

c. Open the LDAP port of the destination IP for each domain controller server by activating the following job in the **Others > Discovery Tools** module:

- **TCP Ports.** This job includes the **TCP_NET_Dis_Port** adapter.

3. **Run the discovery**

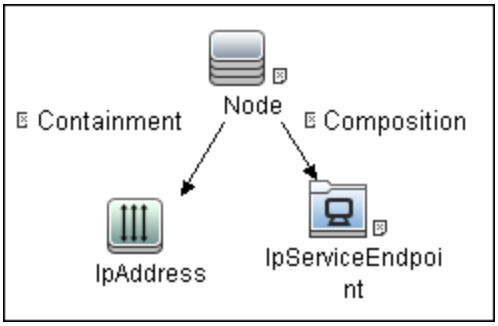
The jobs for AD Discovery are located under **Enterprise Applications > Active Directory**.

- 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 121.

Active Directory Connection by LDAP 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

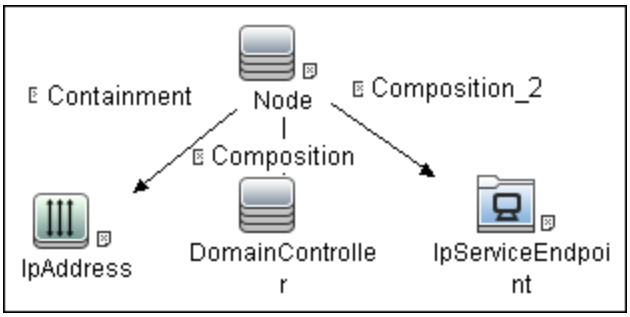
Discovered CITs

- **Containment**
- **Composition**
- **DomainController**
- **Node**
- **IpAddress**

Active Directory Topology by LDAP Job

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.
credentialsId	${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.ip_address}	The IP address of the server.
port	${SERVICE_ADDRESS. ipport_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 116.

Chapter 4

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.

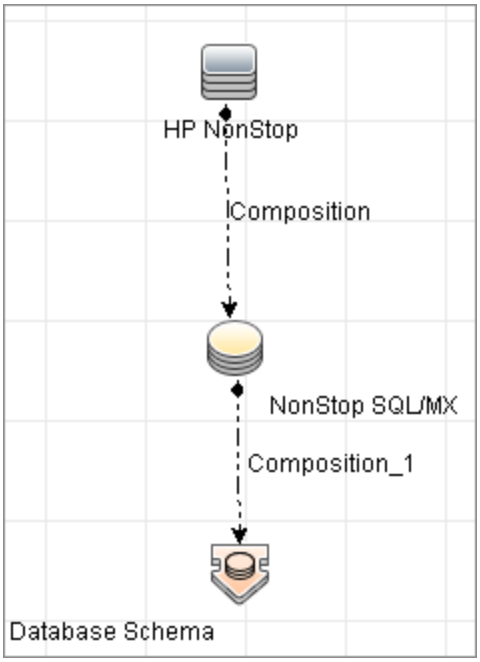
Supported Versions

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



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" on page 82.

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.

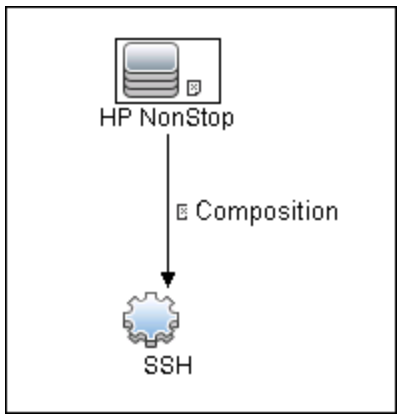
HP NonStop Topology by Shell Job

This section includes:

Trigger Query

The following queries are used for the **HP NonStop Topology by Shell** job:

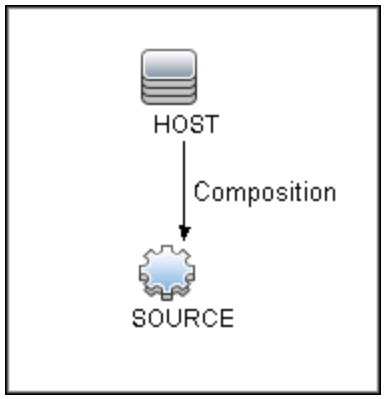
- **Trigger TQL Query**



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	Module	Discovery module
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 this 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
Database instance name	$QA1.H03SQLMP	

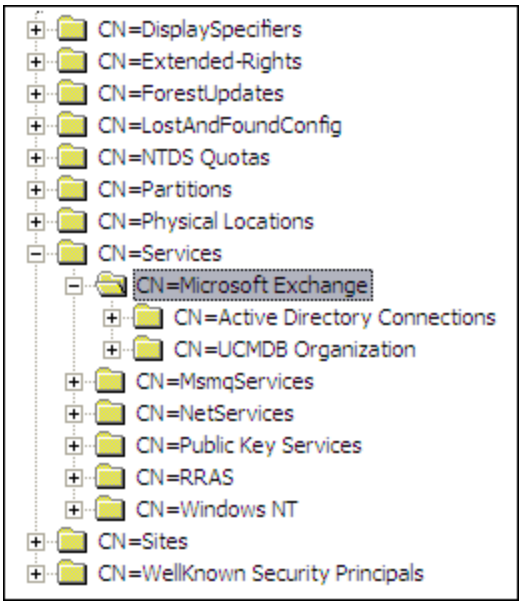
Chapter 5

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:

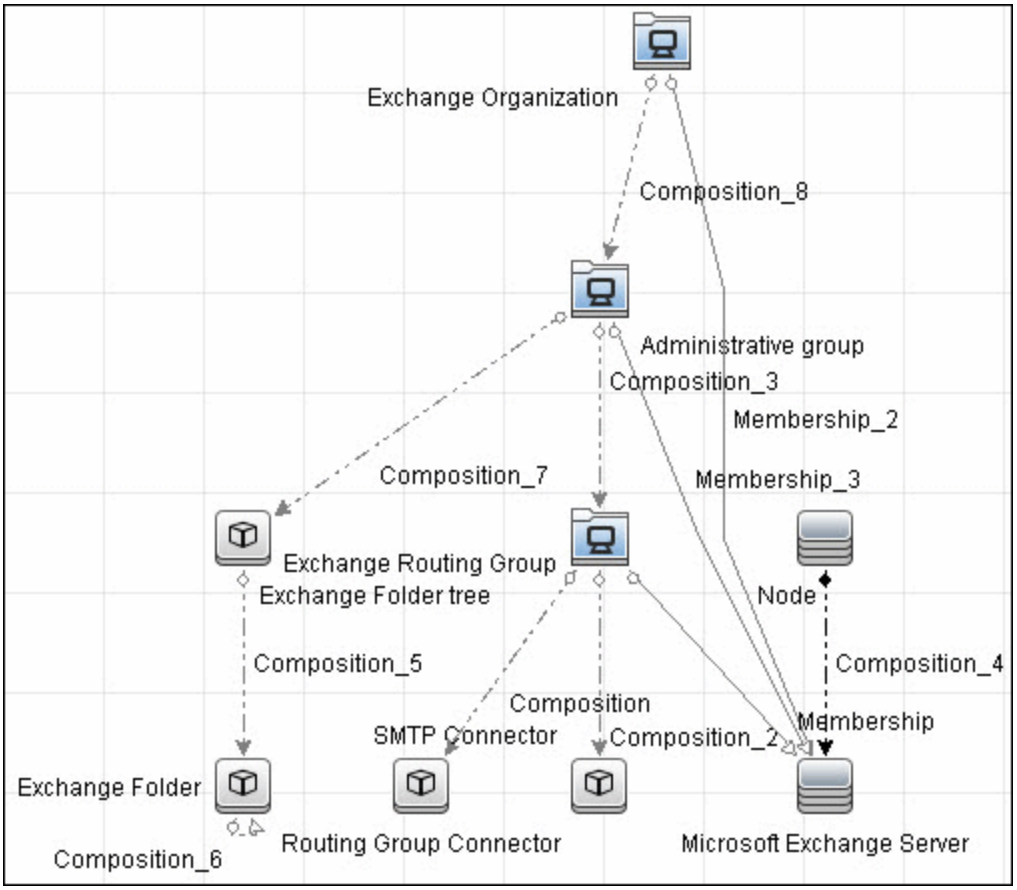
```
Cannot resolve IP address for host '<host>', Exchange Server won't be reported
```

Supported Versions

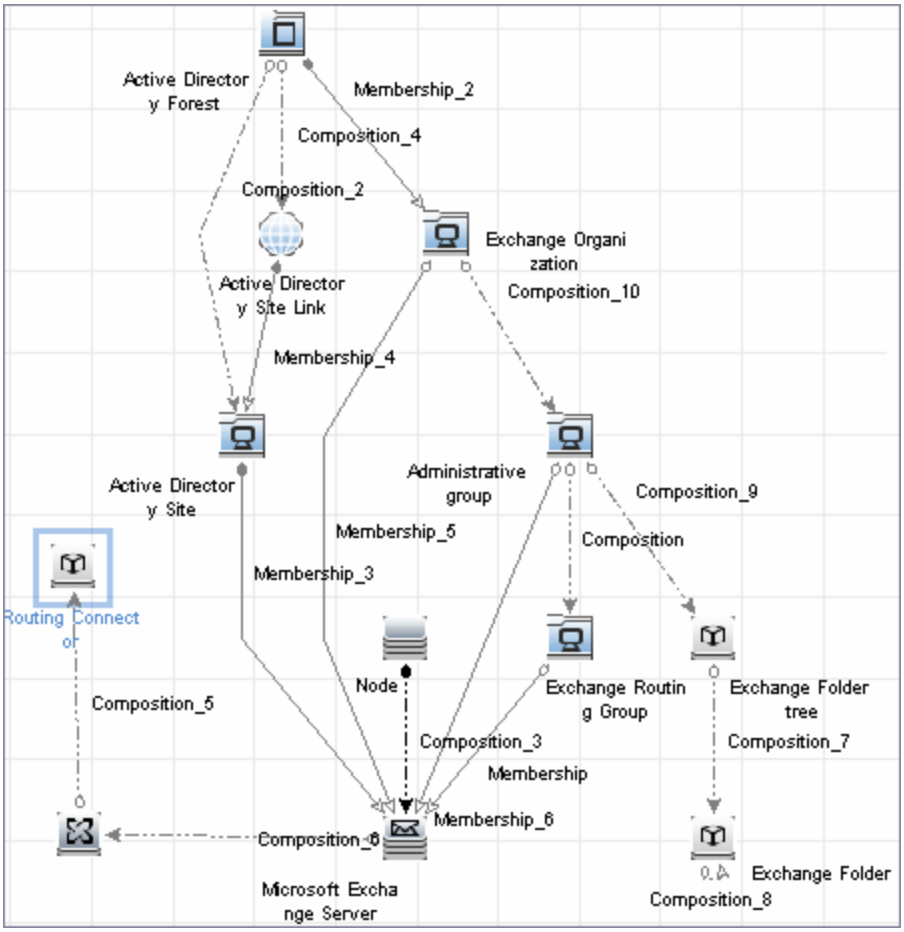
Microsoft Exchange discovery with Active DIrectory supports MS Exchange versions 2003, 2007, and 2010.

Topology

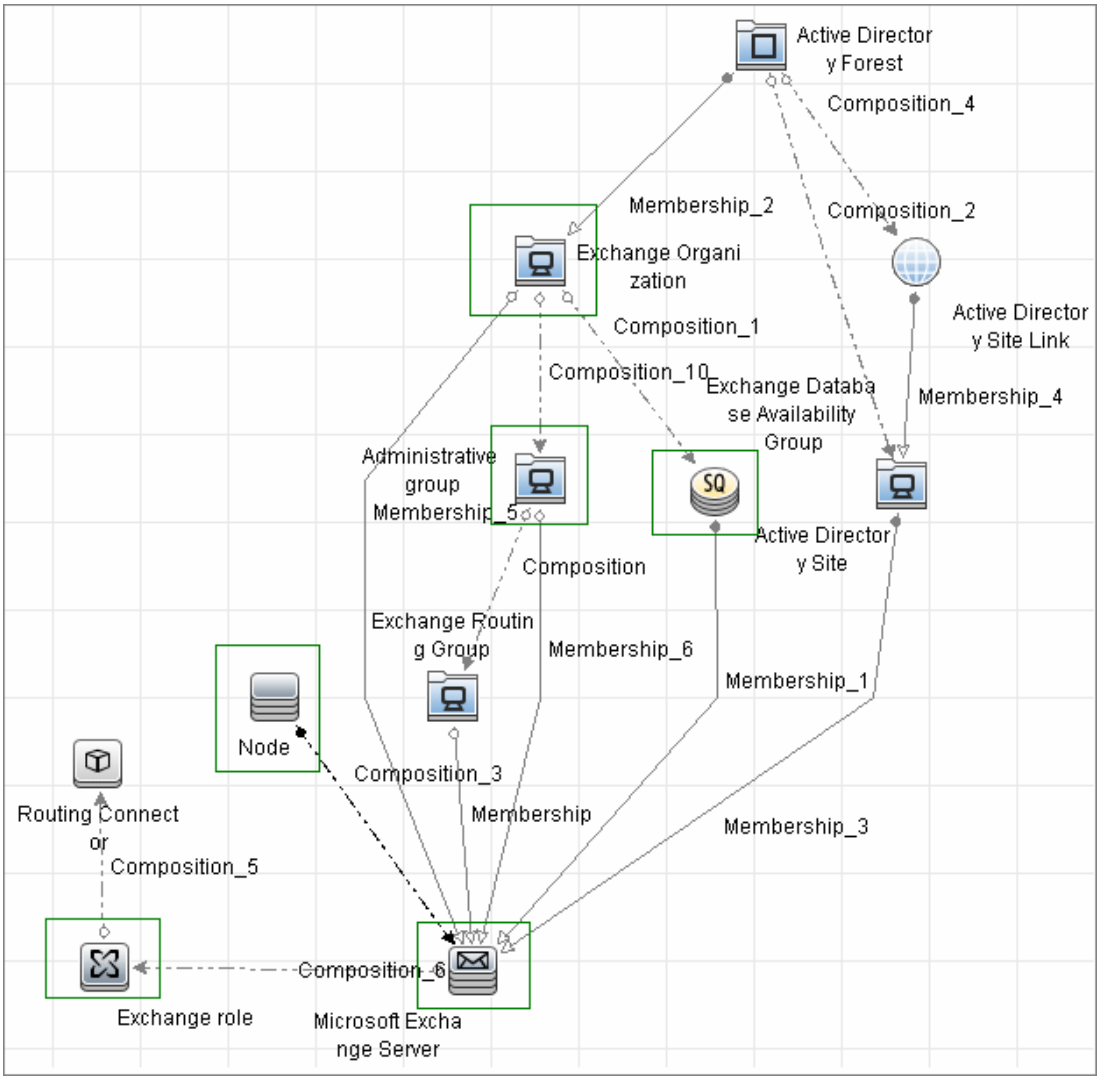
- **Microsoft Exchange Server 2003**



- **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" on page 82.

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.

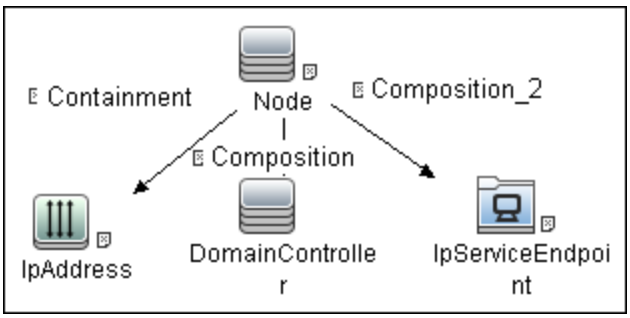
Microsoft Exchange Topology by LDAP Job

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/Changes 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**
- **Containment**
- **Composition**
- **Exchange Database Availability Group**
- **Exchange Folder**
- **Exchange Folder Tree**
- **Exchange Mailbox Database**
- **Exchange Organization**
- **Exchange Routing Connector**
- **Exchange role**
- **ExecutionEnvironment**
- **Host**
- **IpAddress**
- **Membership**
- **Microsoft Exchange Server**
- **Ownership**
- **Routing Group Connector**
- **Exchange Routing group**
- **SMTP Connector**

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Microsoft Exchange Server Topology with Active Directory discovery.

- Currently Exchange Folders are not reported through the **Microsoft Exchange Topology by LDAP** job.

Chapter 6

Microsoft Exchange Server by NTCMD Discovery

This section includes:

Overview

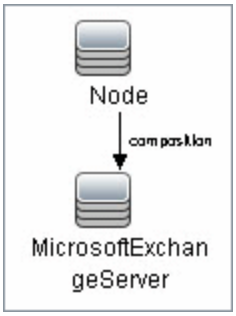
DFM discovers the following components of Microsoft Exchange Server (Exchange) software: Microsoft Exchange Server, Server Roles, Administrative and Routing groups, Organization, Clustered Mail Box, Database Availability group, Public folders, and Folder trees.

Supported Versions

Microsoft Exchange Server by NTCM Discovery supports MS Exchange Server version 2007, 2010.

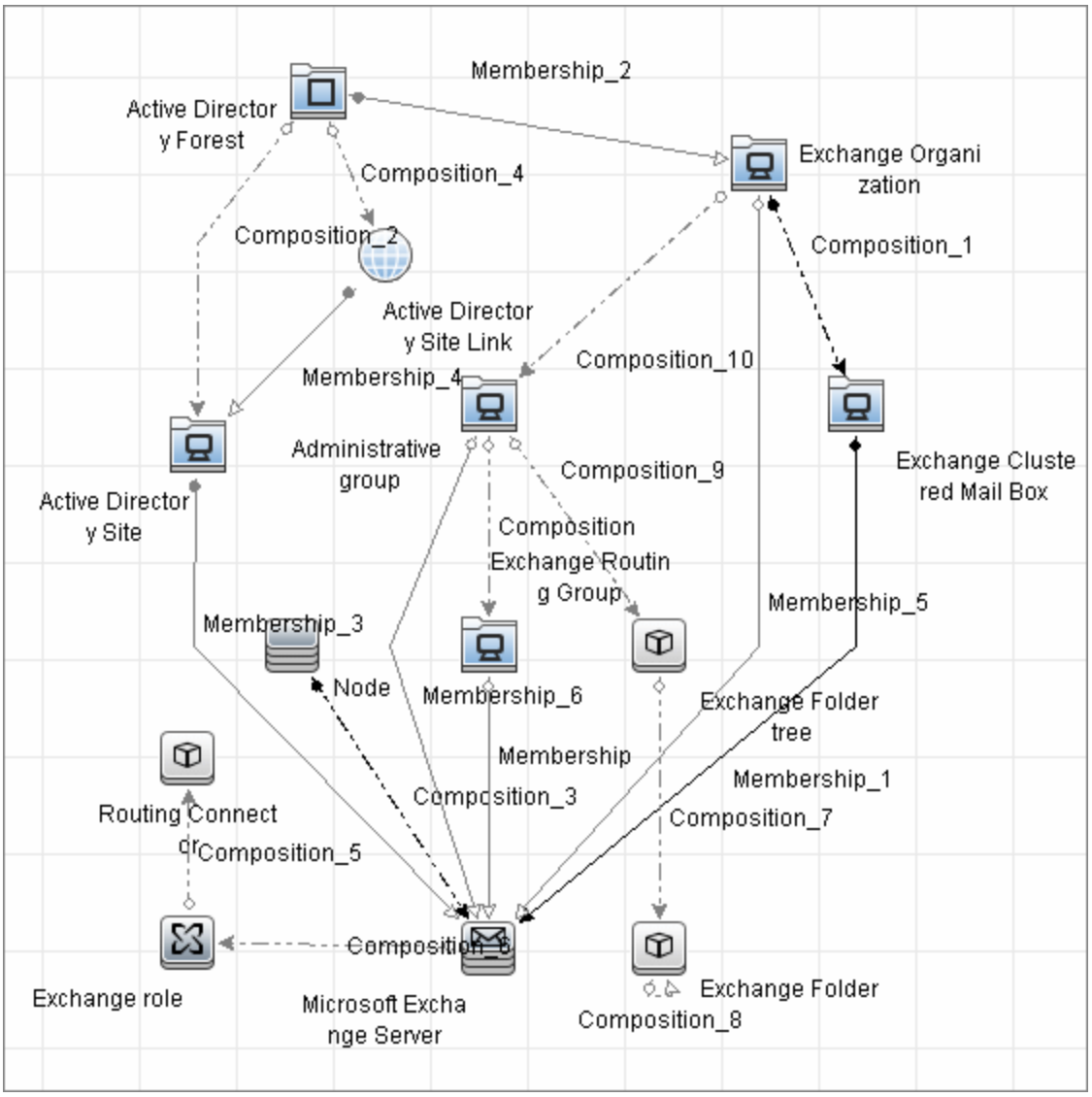
Topology

MS Exchange Connection by NTCMD or UDA:

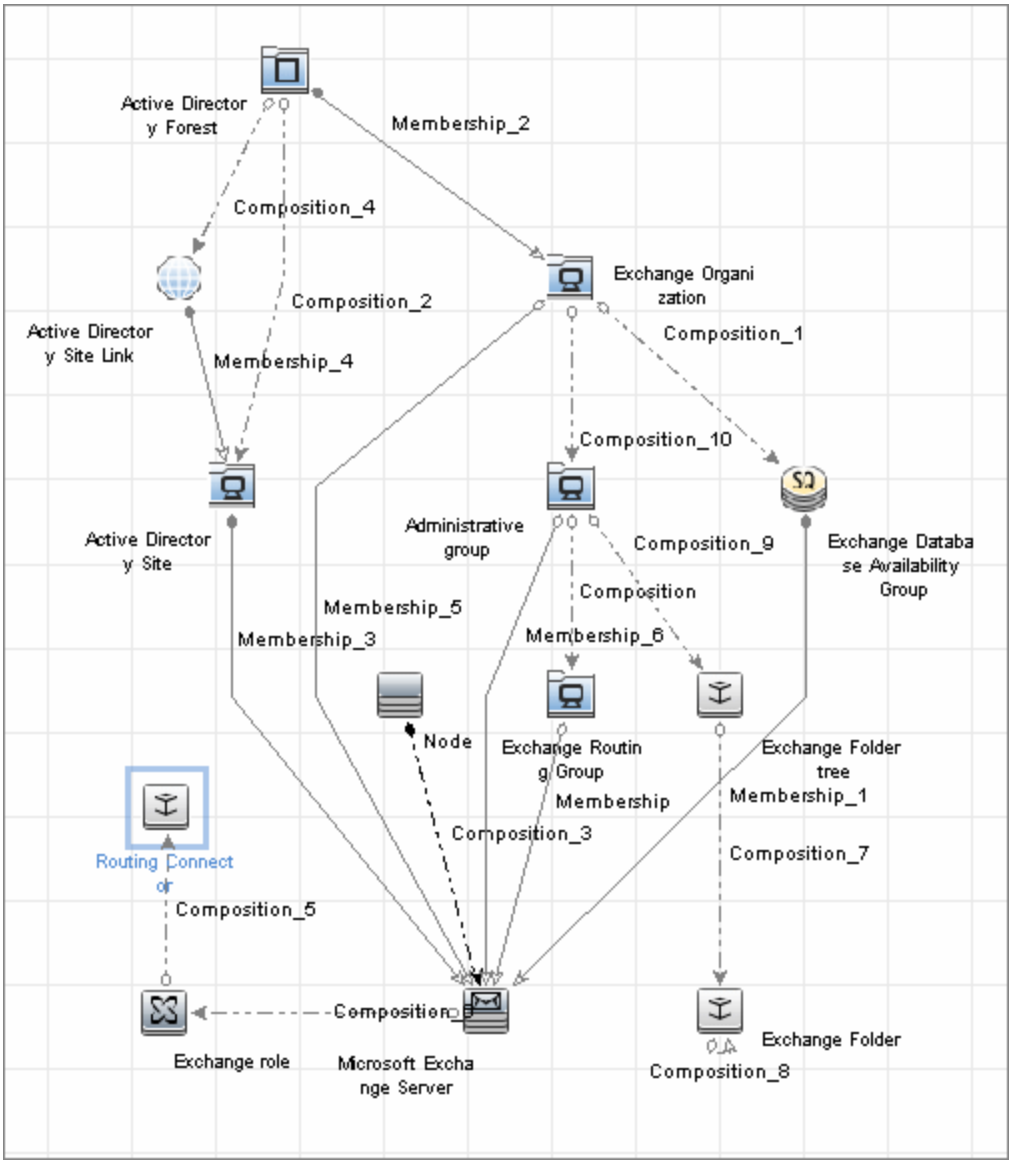


MS Exchange 2007 Topology:

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

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.

   For credential information, see "Supported Protocols" on page 82.

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.

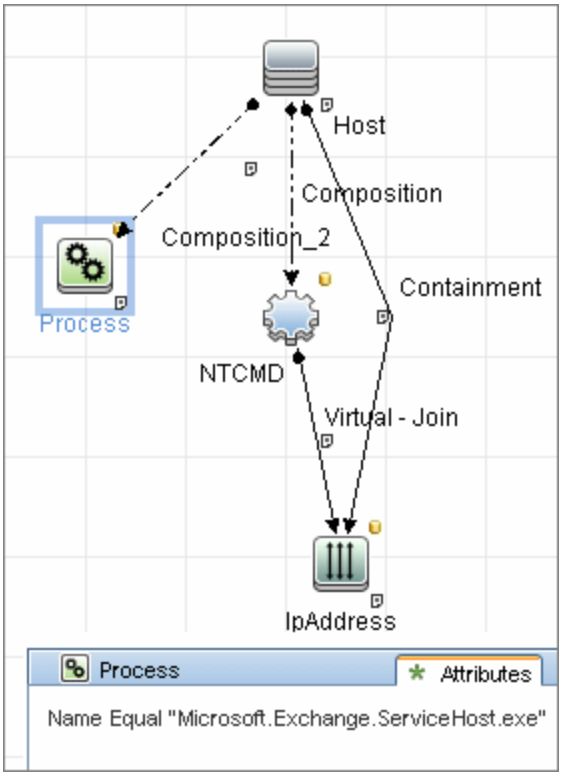
   b. Run the **Host Resources and Applications by Shell** job to discover the Exchange process.

   c. Run the **Microsoft Exchange Connection by NTCMD** job to discover Exchange Server CIs.

   d. Run the **Microsoft Exchange Topology by NTCMD** job to discover the rest of the topology.

Microsoft Exchange Connection by NTCMD Job

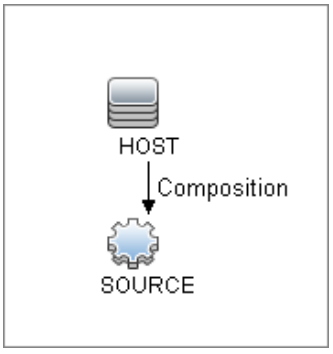
Trigger Query



Adapter

This job uses the **ms_exchange_connection_by_ntcmd** adapter.

- **Input query:**

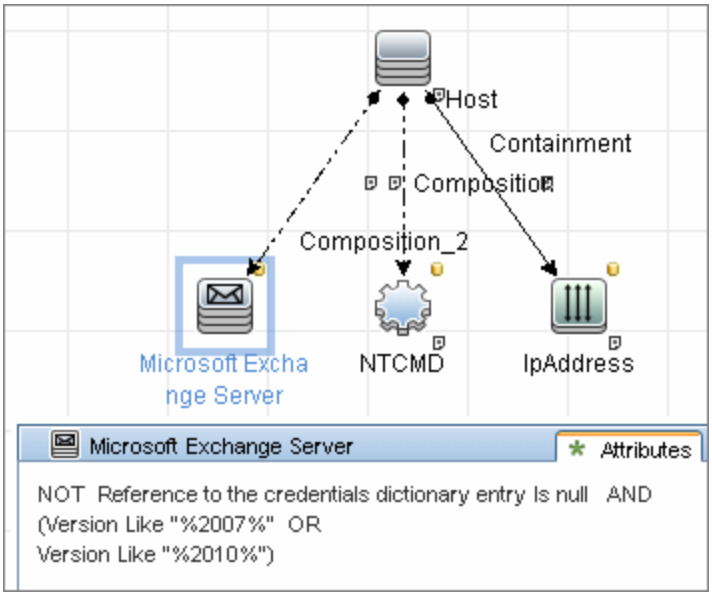


Discovered CITs

- **Composition**
- **MicrosoftExchangeServer**
- **Node**

Microsoft Exchange Topology by NTCMD Job

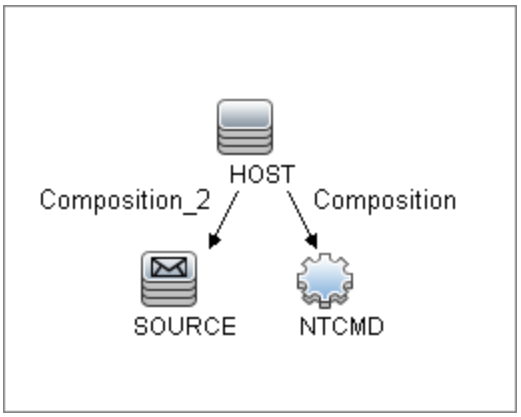
Trigger Query



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 Folder**
- **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 7

Microsoft Exchange Server by PowerShell Discovery

This section 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.

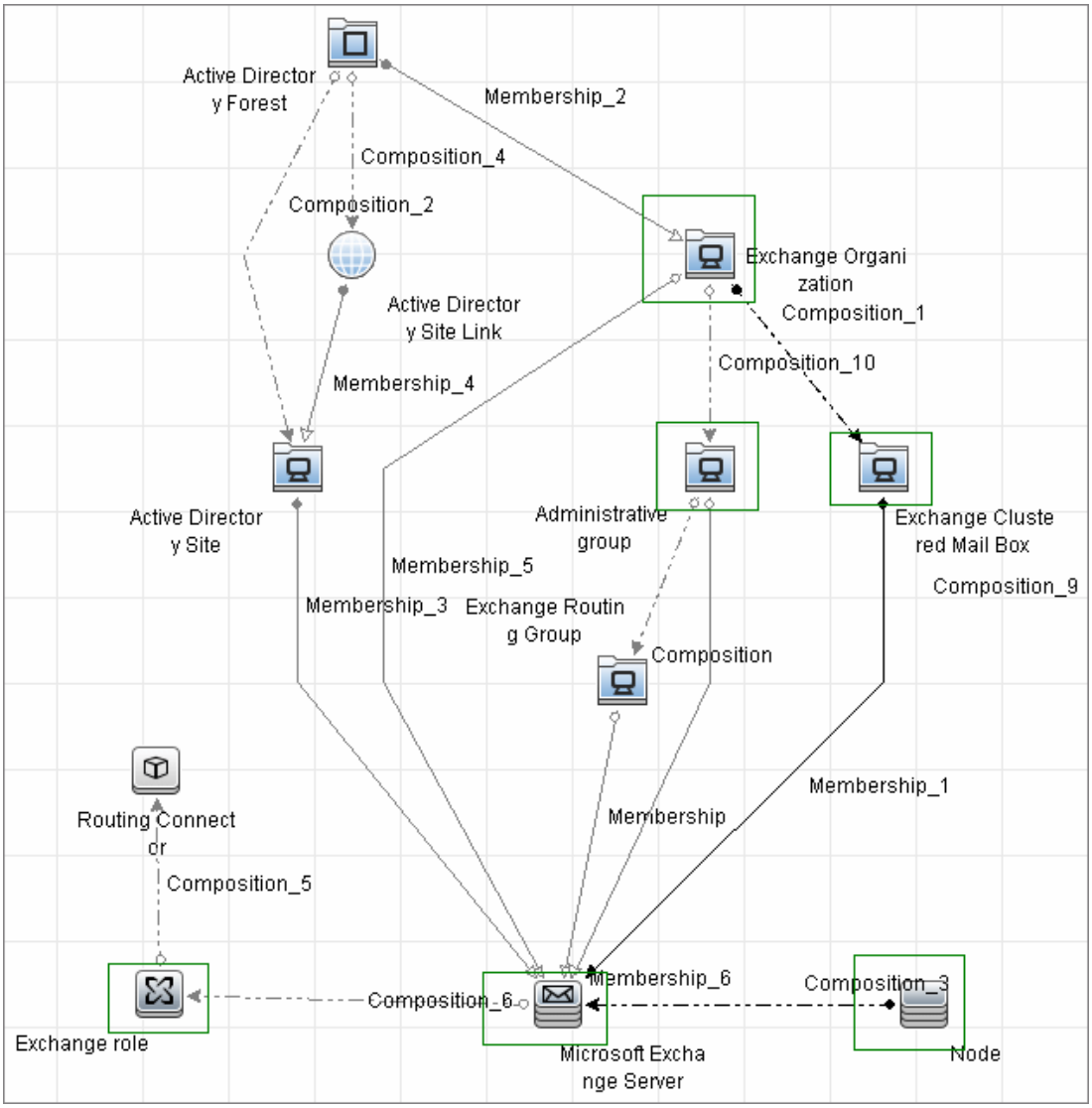
Supported Versions

Microsoft Exchange by PowerShell discovery supports MS Exchange Server versions 2007 and 2010.

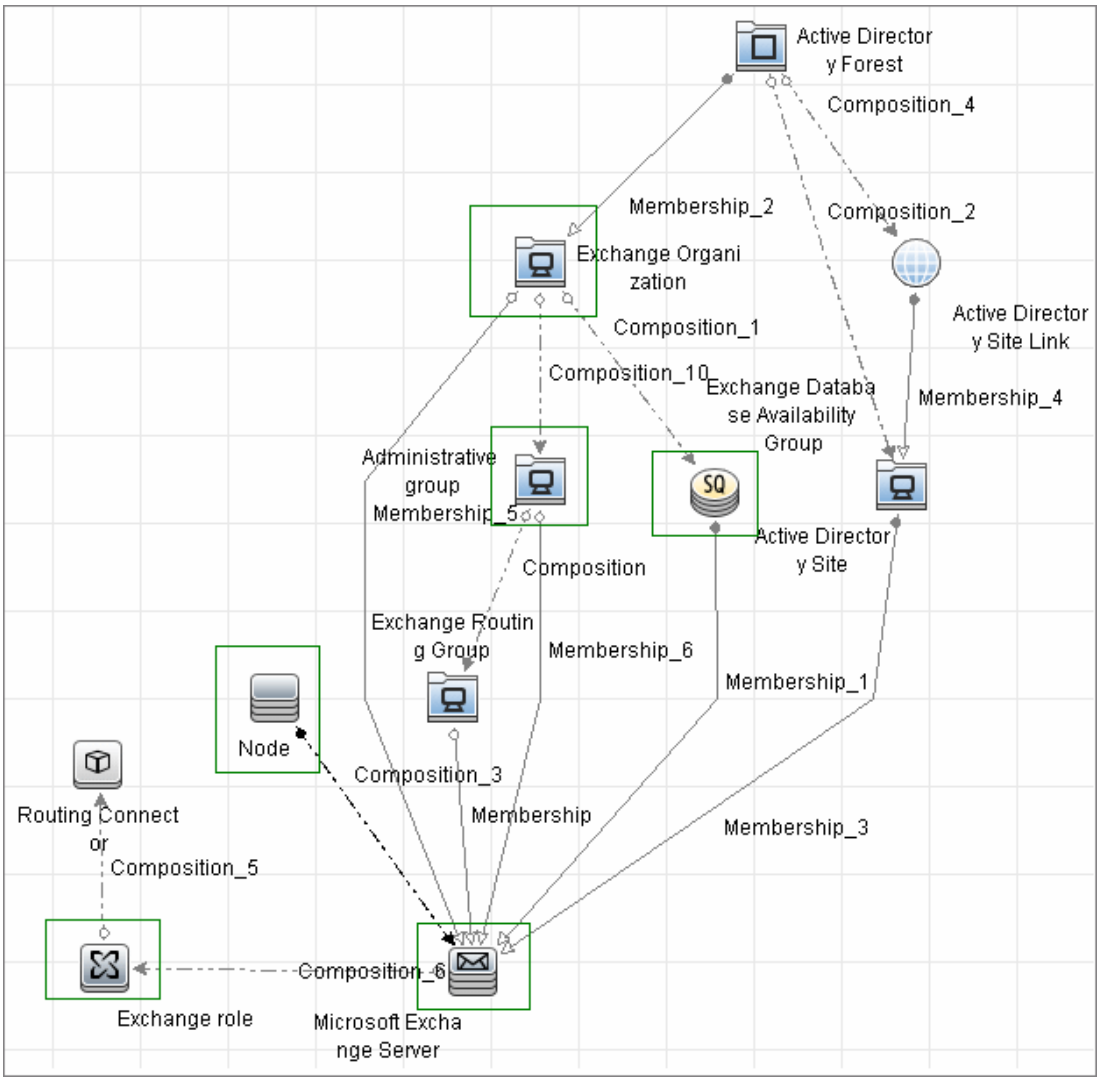
Topology

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 following protocol:

   - PowerShell protocol

   For credential information, see "Supported Protocols" on page 82.

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 next page.

   b. Configure the Active Directory side. For details, see "How to Configure the Active Directory Side" on page 162.

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 Resources and 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.

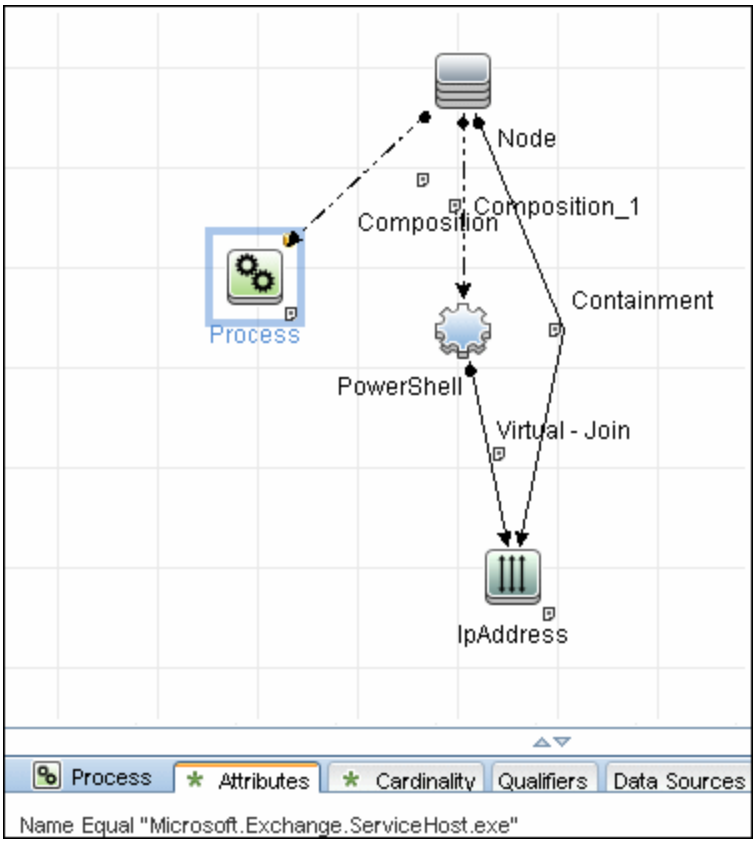
   f. If only the Kerberos protocol is used, select the **Trust this computer for delegation to any service** or **Use Kerberos only**.

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**.

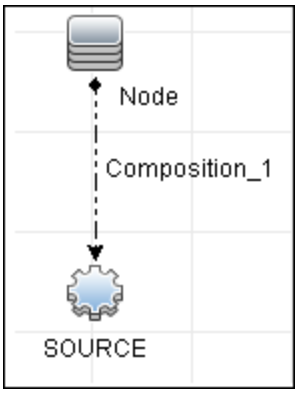
Trigger Query



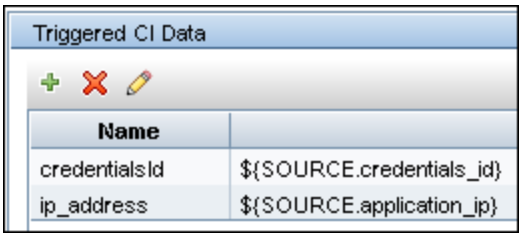
Adapter

This job uses the **MS_Exchange_Topology_by_Powershell** adapter.

- **Input Query**



- **Triggered CI Data**



- **Used Scripts**

  The following scripts are used by Microsoft Exchange by PowerShell discovery.

  - **ms_exchange_topology_by_powershell.py**
  - **ms_exchange_win_shell.py**
  - **ms_exchange.py**
  - **host_win.py**
  - **host_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
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 Mail Server**
- **Exchange Mailbox Database**
- **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 162.

- **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 please see official Microsoft site).

  Solution: Configure Active Directory "Delegation" as described in "How to Configure the Active Directory Side" on page 162, 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 8

Microsoft Exchange Server by WMI Discovery

This section 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

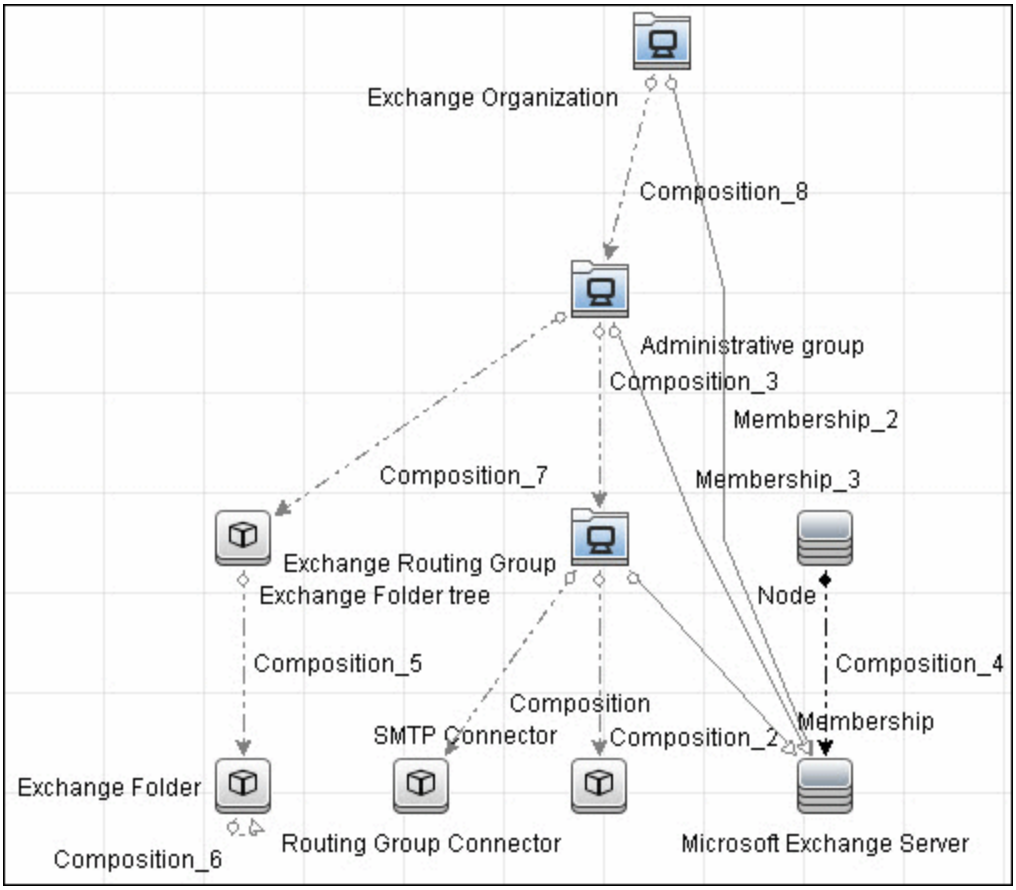
Supported Versions

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" on page 82.

   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 179.

3. **Run the discovery**

   Activate the following jobs:

   - **Network Discovery:**
     - Run **Basic** > **Host Connection by WMI** to discover WMI CITs.
     - 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**
     - 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.
     - 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 179.)

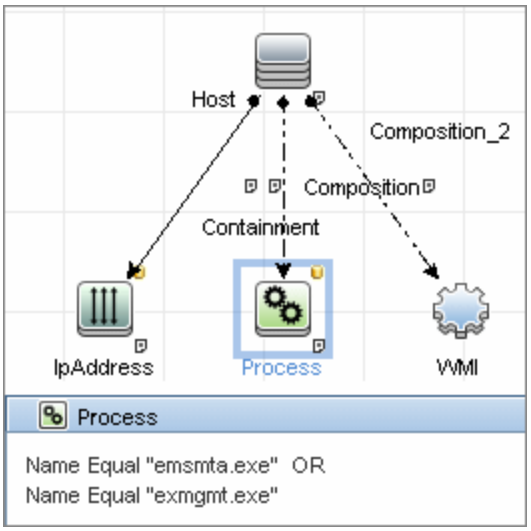
This job connects to the remote host by WMI to the **root\MicrosoftExchangeV2** namespace. The following WMI queries are executed (order is preserved):

```
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

Trigger Query

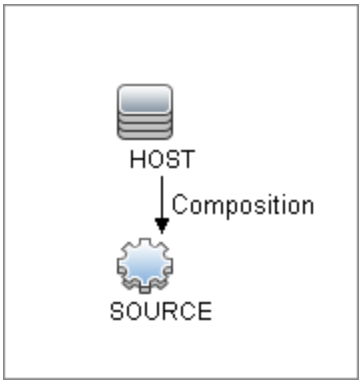
- **Trigger CI:** ms_exchange_process_and_wmi
- **Trigger query:**



Adapter

This job uses the **MS_Exchange_Connection_by_WMI** adapter.

- **Input query:**



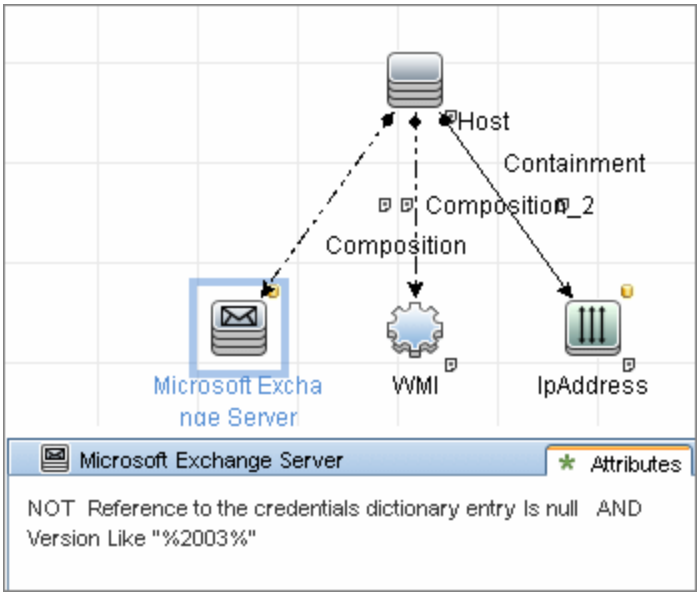
Discovered CITs

- **Composition**
- **Computer**
- **MicrosoftExchangeServer**

Microsoft Exchange Topology by WMI Job

Trigger Query

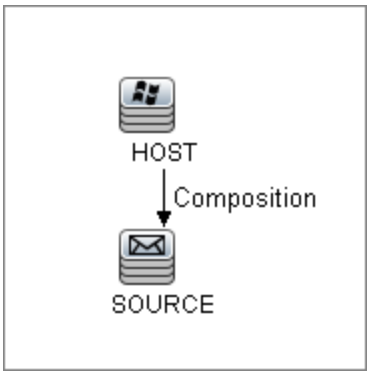
- **Trigger CI:** ms_exchange_server_and_host_and_wmi
- **View:** Microsoft Exchange Topology
- **Trigger query:**



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 173.

Chapter 9

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 **IPs by ICMP** job, or the **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 **IPs by ICMP** job, or the **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 Resources and Applications by Shell** job.
   c. Run the **Host Resources and 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).

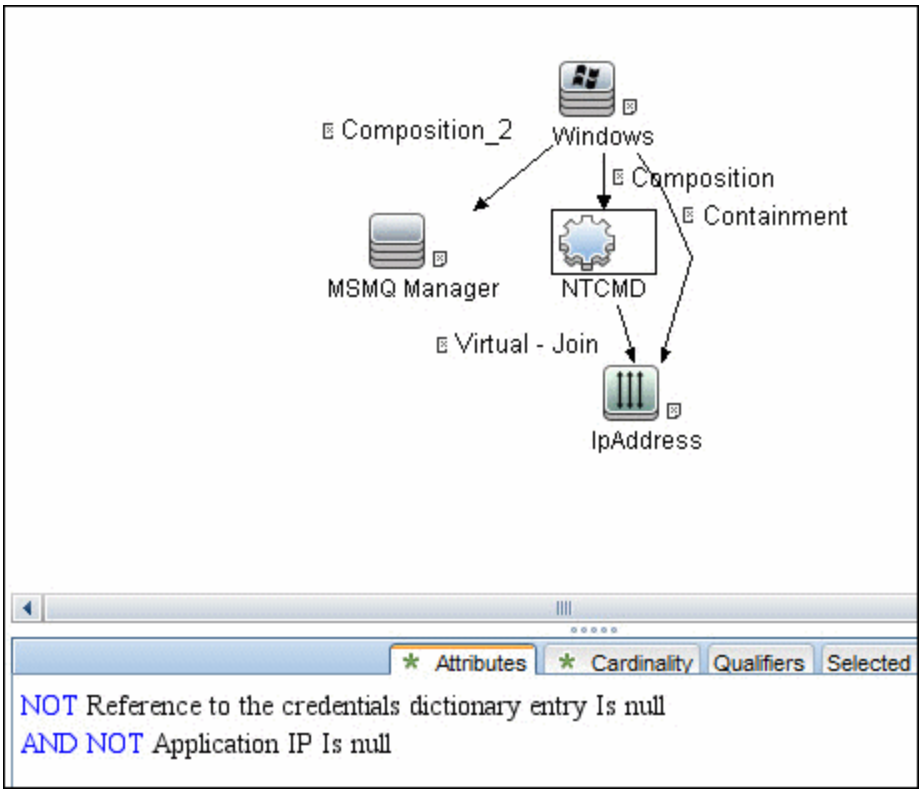
Note: Because information is retrieved from configuration files in three short registry branches only, and each file is less than 2 KB, system performance should not be affected.

For details on how DFM discovers MQ topology, see "Microsoft MQ Topology Discovery Methodology" on page 190.

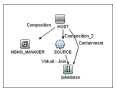
For details on running jobs, refer to "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Microsoft Message Queue Topology by NTCMD Job

Trigger Query



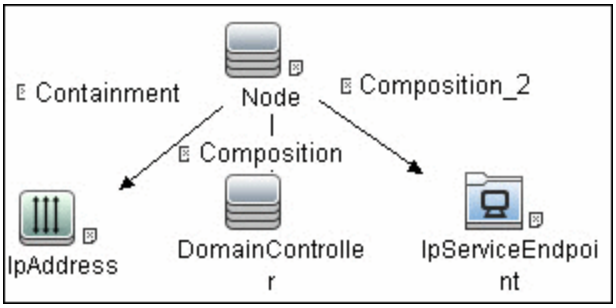
Input Query



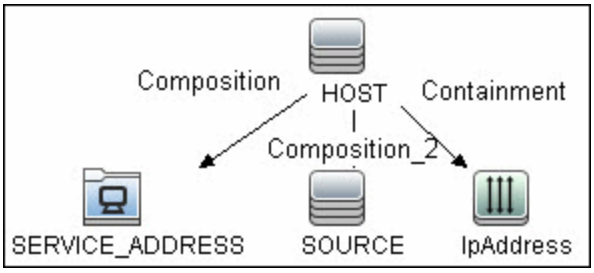
Node Name	Condition
SOURCE	None
HOST	None
MSMQ_MANAGER	None
IpAddress	NOT IP Probe Name Is Null

Microsoft Message Queue Topology by LDAP Job

Trigger Query



Input Query



Microsoft MQ Discovery Scripts

To view the scripts: **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 Discovery Created/Changed Entities

This section includes:

Added Entities

The following entities were added to UCMDB:

Entity Type	Changed Entity
CI Type	Messagingsoftware
CI Type	Mqresource
CI Type	Msmqmanager
CI Type	Msmqqueue
CI Type	Msmqroutinglink
CI Type	Msmqrule
CI Type	Msmqtrigger
Attribute type definition	MessageProcessingTypeEnum
Type definition	MsMqManagerInstallationType
Type definition	MsMqQueueTypeEnum
Link	clientserver.msmqmanager.msmqmanager
Link	containment.msmqroutinglink.mqqueuemanager
Link	containment.msmqroutinglink.msmqmanager
Link	composition.activedirectoryforest.msmqroutinglink
Link	composition.msmqqueue.msmqtrigger
Link	membership.msmqroutinglink.activedirectorysite
Link	usage.msmqtrigger.msmqrule
Job	Microsoft Message Queue Topology by LDAP
Job	Microsoft Message Queue Topology by NTCMD or UDA

Deprecated Entities

In UCMDB 9.01, the MQ (Microsoft Message Queue) model was changed and the following resources are no longer available:

CIT	Display Name
mqaliasq	IBM MQ Queue Alias
mqalias	IBM MQ Alias
mqchannelof	IBM MQ Channel Of
mqchannel	IBM MQ Channel
mqchclntconn	IBM MQ Client Connection Channel
mqchclusrcvr	IBM MQ Cluster Receiver Channel
mqchclussdr	IBM MQ Cluster Sender Channel
mqchrcvr	IBM MQ Receiver Channel
mqchrqstr	IBM MQ Requester Channel
mqchsdr	IBM MQ Sender Channel
mqchsvrconn	IBM MQ Server Connection Channel
mqchsvr	IBM MQ Sender Channel
mqcluster	IBM MQ Cluster
mqmqichannel	IBM MQ MQI Channel
mqmqilink	IBM MQ
mqmsgchannel	IBM MQ Message Channel
mqmsglink	IBM MQ Message
mqmsgreceiverchannel	IBM MQ Message Receiver Channel
mqmsgsenderchannel	IBM MQ Messenger Sender Channel
mqqueuelocal	IBM MQ Local Queue
mqqueuemanager	IBM MQ Queue Manager
mqqueueremote	IBM MQ Remote Queue
mqqueue	IBM MQ Queue
mqrepository	IBM MQ Repository

CIT	Display Name
mqresolve	IBM MQ Resolve
mqxmitq	IBM MQ Transmission Queue
webspheremq	IBM WebSphere MQ

Removed Entities

The following resources were removed:

Entity Type	Removed Entity
Enrichment rule	Create_Msg_Channel_Link_Host
Enrichment rule	Create_Msg_Channel_Link_IP
Enrichment rule	Create_RemoteQueue_Link
Enrichment rule	Host_Depend_By_MQ
View	MQ_All_Objects
View	MQ_Channels
View	MQ_Clusters
View	MQ_Network_Objects
View	MQ Queue Map
TQLs	All TQLs corresponding to the above Enrichment rules and Views

Microsoft MQ Topology Discovery Methodology

This section describes how DFM discovers the MS MQ topology.

This section includes the following topics:

Host Resources and 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 1050000000000051500000057 6A62162631895
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 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=010007805c000000680000000000000140000000200
480003000000000018003f000e000102000000000052000000020
```

```
020000000014002400020001010000000000010000000000000140
004000000010100000000000507000000010100000000000051200
00000101000000000000512000000
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.

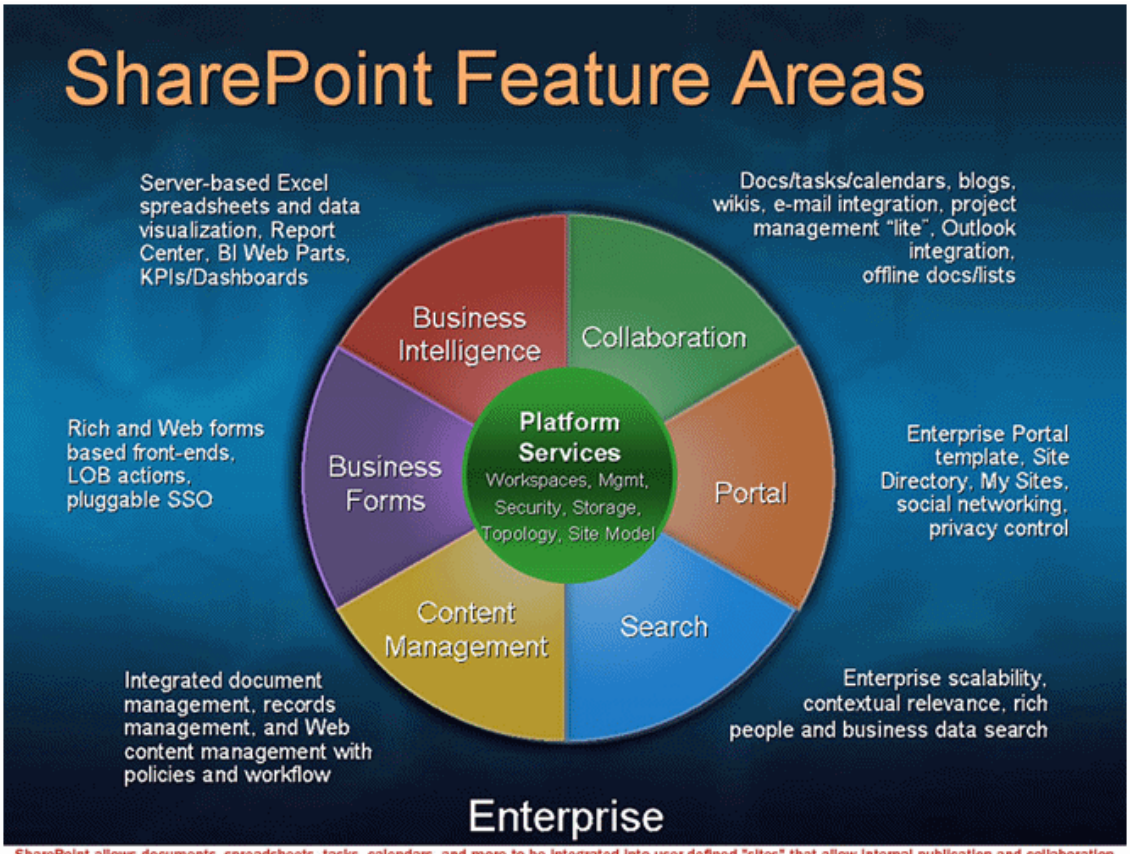
Chapter 10

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 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:

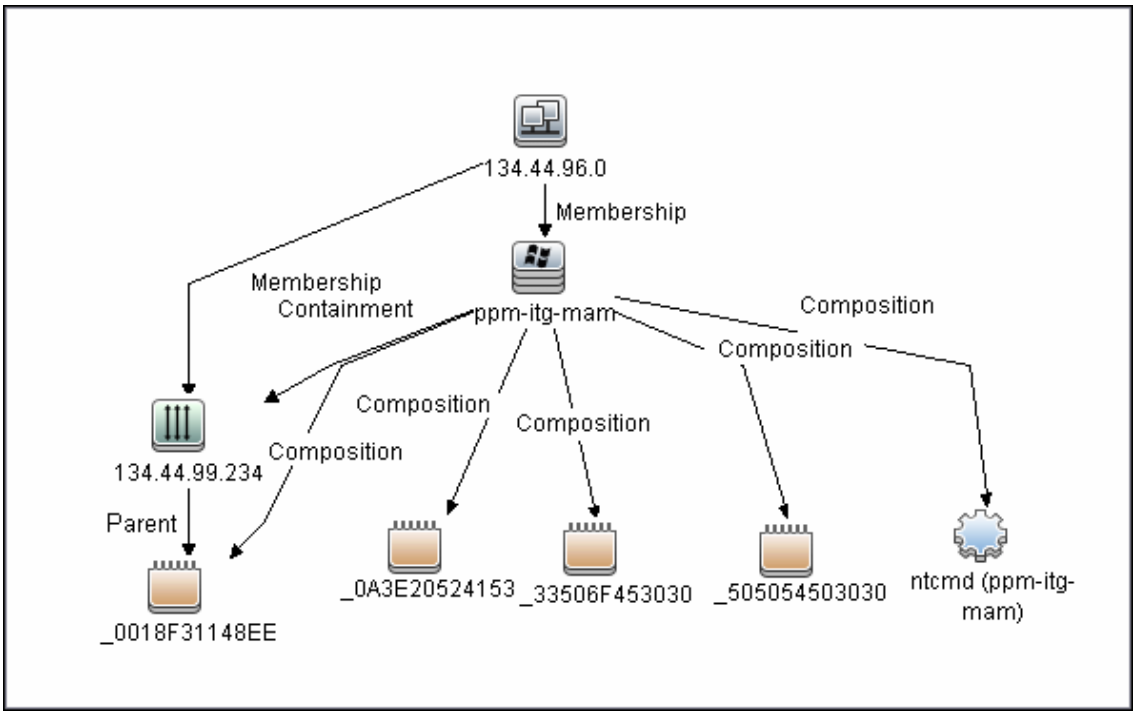
- Microsoft SharePoint 2007
- Microsoft SharePoint Server 2010

Note: This discovery is expected to work on all available versions of Microsoft SharePoint.

Topology

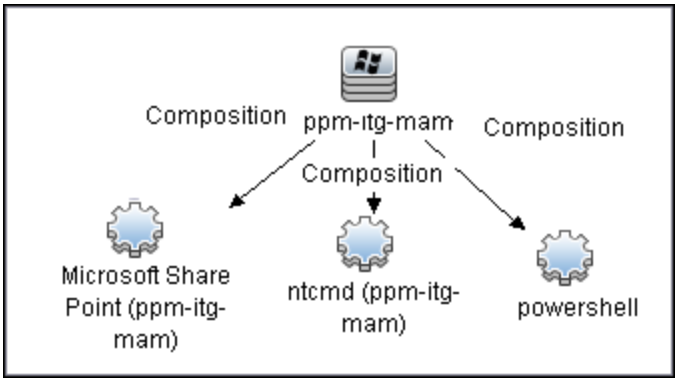
The following images display sample output for the Sharepoint discovery jobs.

Host Connection by Shell Job



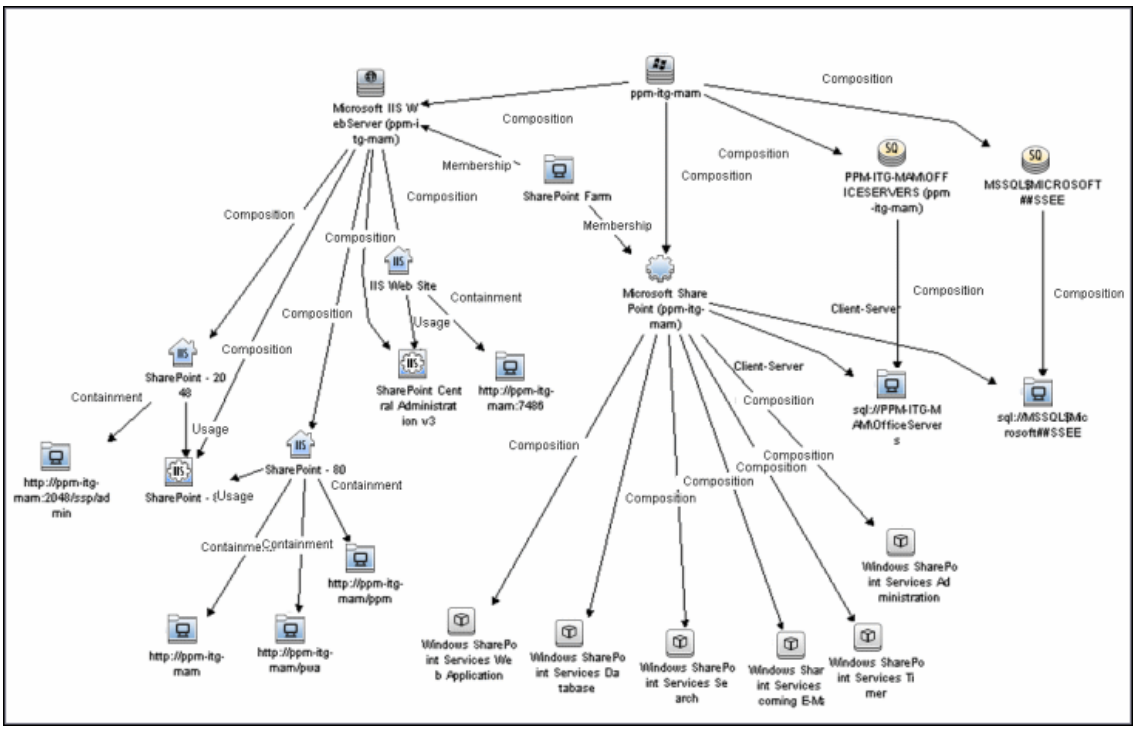
Host Resources and Applications by Shell Job

Note: Only the data necessary for the continued flow is shown.



Microsoft SharePoint Topology Job

Note: For a list of discovered CITs, see "Discovered CITs" on page 207.



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" on page 82.

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.

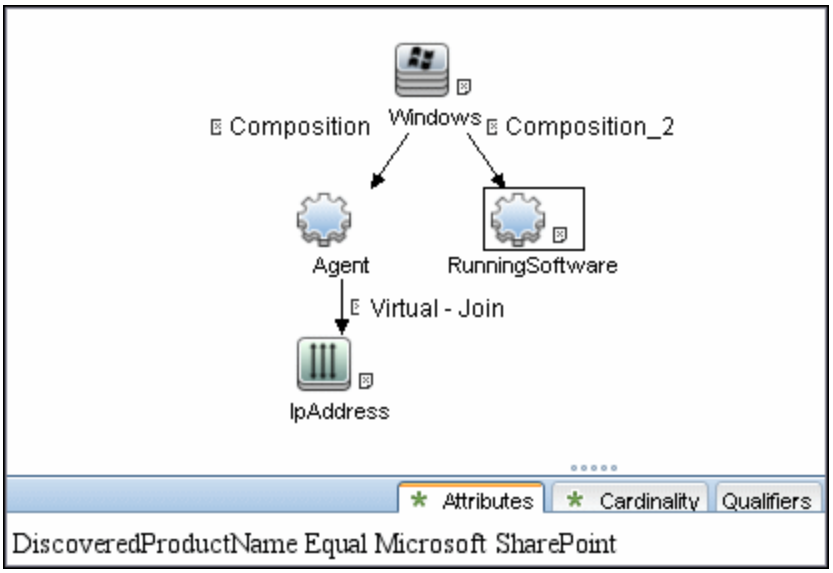
c. Run the **Host Resources and Applications by Shell** or **Host Resources and Applications by PowerShell** job to discover the connection between the SharePoint system and the SharePoint software element, and the detailed host topology.

d. Run the **Microsoft SharePoint Topology** job to discover the Microsoft SharePoint Server topology.

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

Microsoft SharePoint Topology Job

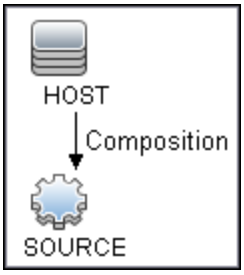
Trigger Query



Note: On IPAddress, the **IP Probe name is not null** attribute is set.

Adapter

- **Input CIT:** Agent
- **Input Query**



- **Used Scripts**
  - sharepointdiscoverer.py
  - sharepoint.py
  - SharePointMain.py

> **Note:** This job may also use library scripts supplied with the Auto Discovery Content package.

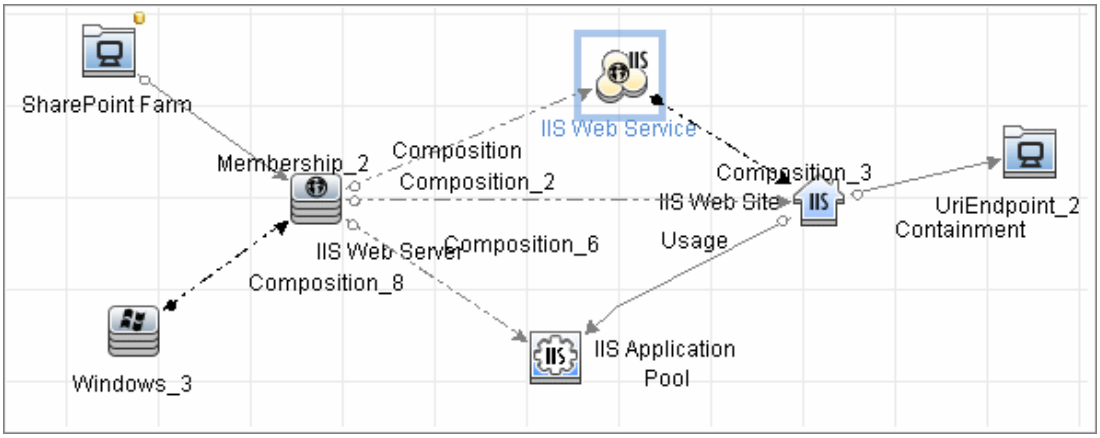
Job Parameters

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:

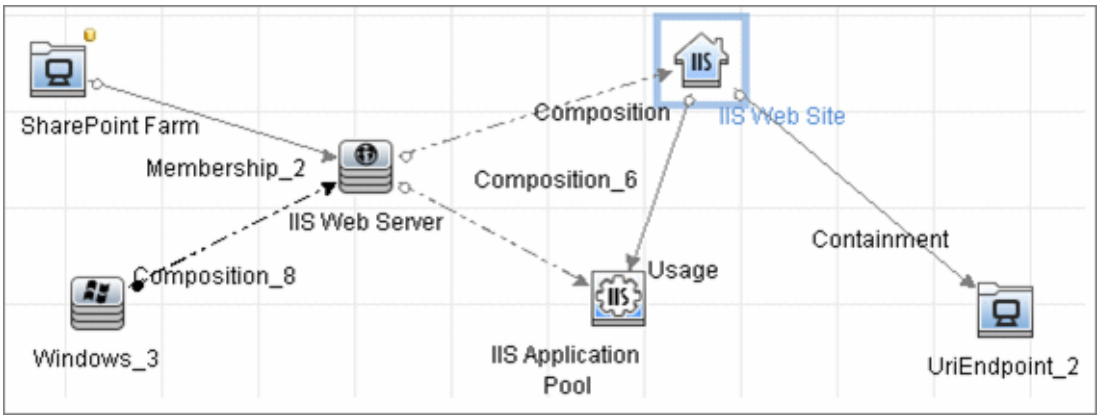
- **reportIntermediateWebService = true**:

  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
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

- **Composition**
- **Containment**
- **IIS Application Pool**
- **IIS Web Server**
- **IIS Web Site**
- **IpAddress**
- **Membership**
- **Running Software**
- **SQL Server**
- **SharePoint Farm**
- **SharePoint Service**
- **UriEndPoint**
- **Usage**
- **Windows**

Note: To view the topology, see "Topology" on page 201.

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

Attribute	Value
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 208 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.

- 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
- For each SharePoint service, all the configuration details are merged into one string in the **service configuration** attribute of the SharePoint Service CIT.
- 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.

Chapter 11

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.

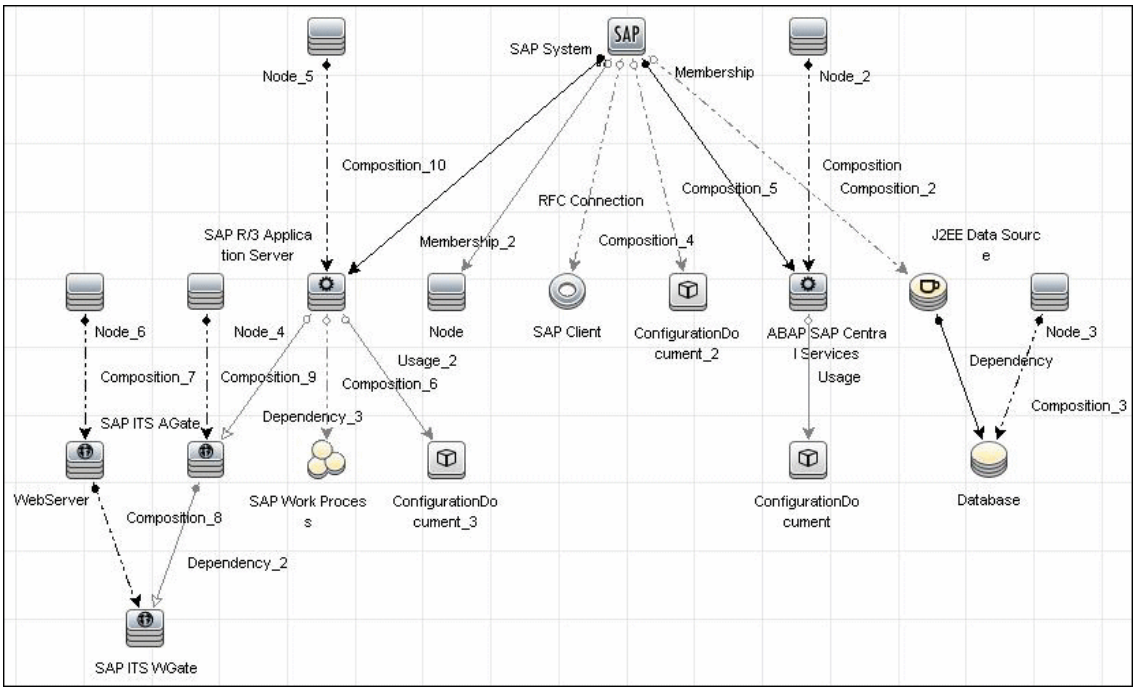
Note: You can discover the whole SAP system by discovering a connection to the SAP Solution Manager. In this way, you create a single set of credentials; there is no need to create a set of credentials for each SAP system. DFM discovers all systems (and their topology) with this one set. For details, see "SAP Solution Manager Discovery" on page 237

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.

This task 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" on page 82.

2. ## Prerequisite – Install Java Connectors

   Note: All actions in this part should be performed on the machine where the Data Flow Probe is installed.

<table>
<tr><th>JCo version 3.x (from 3.0.7)</th><th>JCo version 2.x</th></tr>
<tr><td colspan="2">- Download the SAP JCo package. This is accessible from the SAP Service Marketplace > SAP JCo >Tools & Services window: http://service.sap.com/connectors</td></tr>
<tr><td colspan="2">- Extract the JCo installation ZIP content to a temporary directory (for example: C:\temp).</td></tr>
<tr><td>- Copy sapjco3.jar and sapjco3.dll from the temporary directory to the <DataFlowProbe_root>\content\lib\ directory.</td><td>- Copy sapjco.jar and sapjcorfc.dll from the temporary directory to the <DataFlowProbe_root>\content\lib\ directory.
- 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.</td></tr>
</table>

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**

Discovery	Configuration
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 Discovery Control Panel window, activate the jobs in the following order:

      For details on running jobs, see "Discovery Control Panel" 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.**
      - **WebServer Detection using TCP Ports**. If the SAP system has an ITS configuration, to discover the ITS entities of the SAP system, run this job as a prerequisite to the SAP discovery that discovers ITS entities.
      - **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 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 237.

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*.

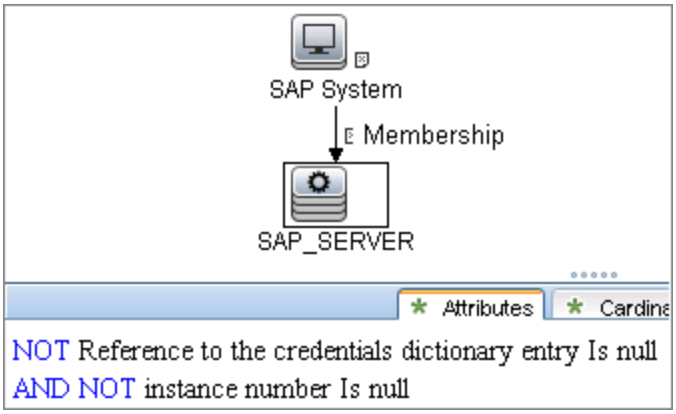
c. Verify that DFM discovered the appropriate components. Access the **SAP_ABAP_ Topology** view in the Modeling Studio and verify that the map displays all components.

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

Trigger Query

- **Trigger CI: SAP ABAP Application Server**



Used Scripts

- **sapapputils.py**
- **saputils.py**
- **sap_solution_topology.py**

Discovered CITs

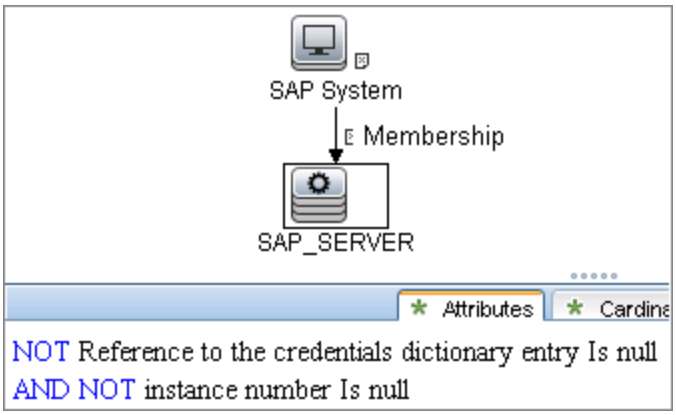
- **ABAP SAP Central Services**
- **Composition**
- **Configuration Document**
- **Containment**
- **Database**
- **Dependency**
- **IpAddress**
- **J2EE SAP Central Services**

- **JDBC Data Source**
- **Membership**
- **Node**
- **SAP ABAP Application Server**
- **SAP Client**
- **SAP J2EE Application Server**
- **SAP System**
- **Usage**

SAP Solution Manager by SAP JCO Job

Trigger Query

- **Trigger CI: SAP ABAP Application Server**
- **Trigger query:**



Used Scripts

- **sapapputils.py**
- **saputils.py**
- **sap_solution_manager.py**

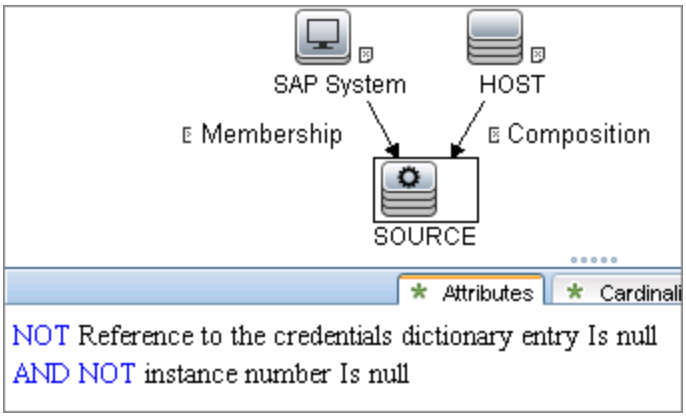
Discovered CITs

- **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

Trigger Query

- **Trigger CI: SAP ABAP Application Server**
- **Trigger query:**



Used Scripts

- **sapapputils.py**
- **saputils.py**
- **sap_applications.py**

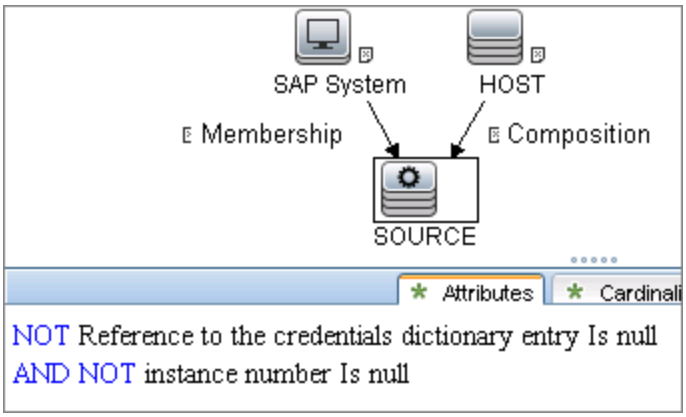
Discovered CITs

- **Composition**
- **Containment**
- **SAP Application Component**
- **SAP System**
- **SAP Transaction**
- **SAP Transport**
- **SAP Transport Change**
- **Usage**

SAP ABAP Topology by SAP JCO Job

Trigger Query

- **Trigger CI: SAP ABAP Application Server**
- **Trigger query:**



Used Scripts

- **sap.py**
- **sapapputils.py**
- **saputils.py**

Discovered CITs

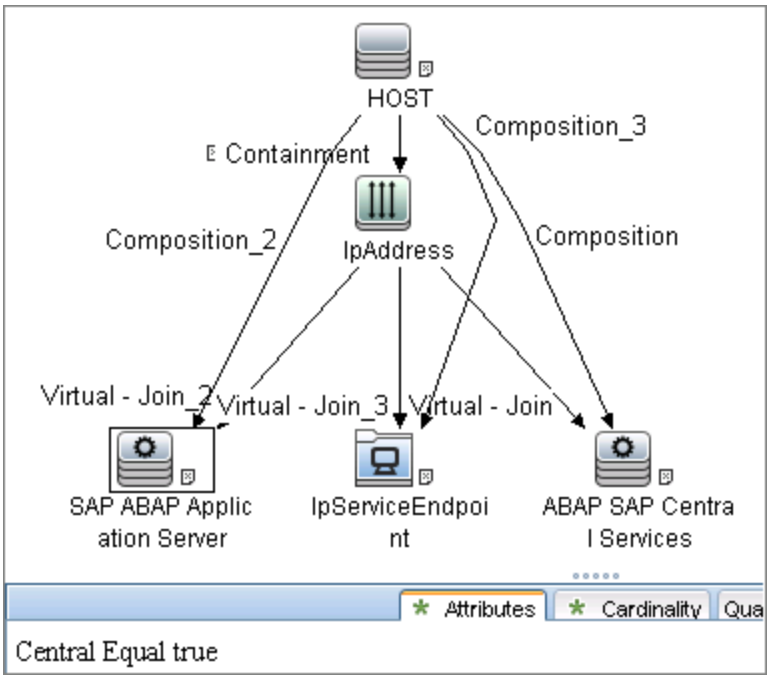
- **Composition**
- **ConfigurationDocument**
- **Containment**
- **Database**
- **Dependency**
- **IPAddress**
- **JDBC Data Source**
- **Membership**
- **Node**
- **RFC Connection**
- **RunningSoftware**
- **SAP ABAP Application Server**
- **SAP Client**
- **SAP Gateway**

- **SAP System**
- **SAP Work Process**
- **Usage**

SAP ABAP Connection by SAP JCO Job

Trigger Query

- **Trigger CI: IpAddress**
- **Trigger query:**



Used Scripts

- **sapapputils.py**
- **saputils.py**
- **sap_system_dis.py**

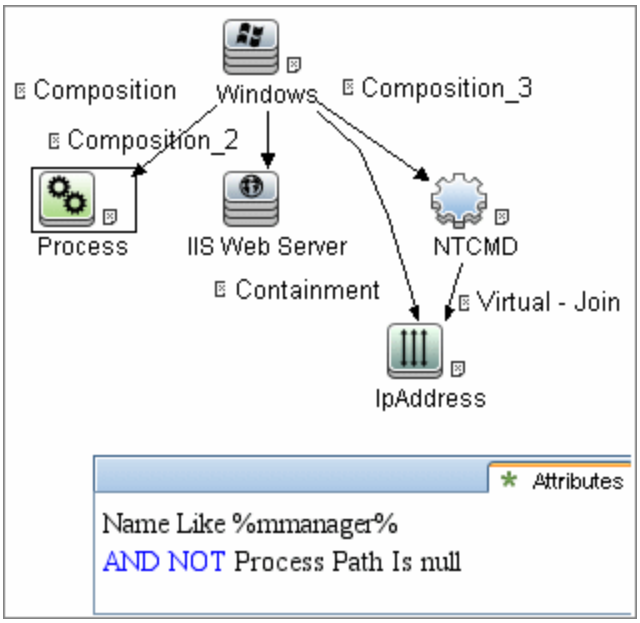
Discovered CITs

- **Composition**
- **Containment**
- **IPAddress**
- **Membership**
- **Node**
- **SAP ABAP Application Server**
- **SAP System**

SAP ITS by NTCMD or UDA Job

Trigger Query

- **Trigger CI: IIS Web Server**
- **Trigger query:**



Used Script

- **sap_its.py**

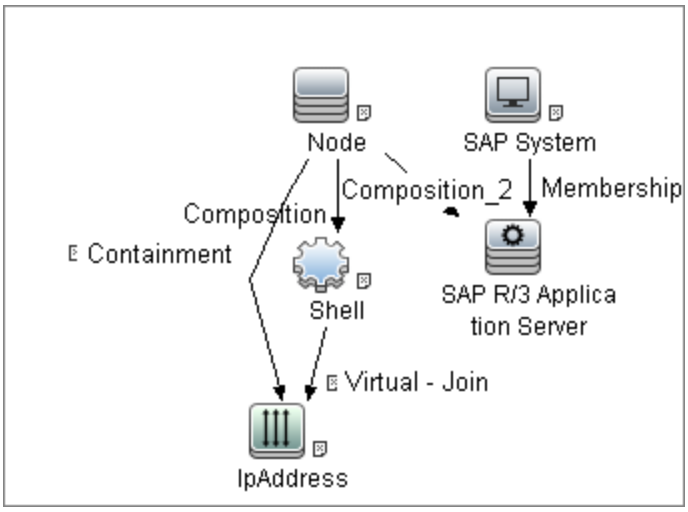
Discovered CITs

- **Composition**
- **Containment**
- **Dependency**
- **IPAddress**
- **Node**
- **SAP ABAP Application Server**
- **SAP ITS AGate**
- **SAP ITS WGate**
- **WebServer**

SAP Profiles by Shell Job

Trigger Query

- **Trigger CI: SapApplicationServer**
- **Trigger query:**



Used Scripts

- **file_mon_utils.py**
- **file_ver_lib.py**
- **sap_profiles_by_shell.py**

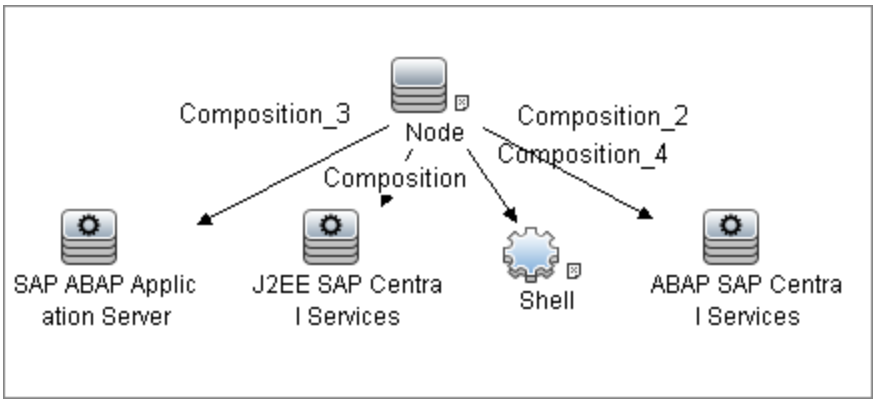
Discovered CITs

- **Composition**
- **ConfigurationDocument**
- **Usage**

SAP System By Shell Job

Trigger Query

- **Trigger CI: SapApplicationServer**
- **Trigger query:**



Used Scripts

- **file_mon_utils.py**
- **file_ver_lib.py**
- **sap_profiles_by_shell.py**
- **sap_system_by_shell.py**

Discovered CITs

- **Composition**
- **ConfigurationDocument**
- **Membership**
- **SAP System**
- **SapApplicationServer**
- **Usage**

SAP TCP Ports Job

Trigger Query

- **Trigger CI: IpAddress**
- **Trigger query:**

Used Script

- **TcpPortScanner.py**

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.

Chapter 12

SAP Java Discovery

This chapter includes:

Overview

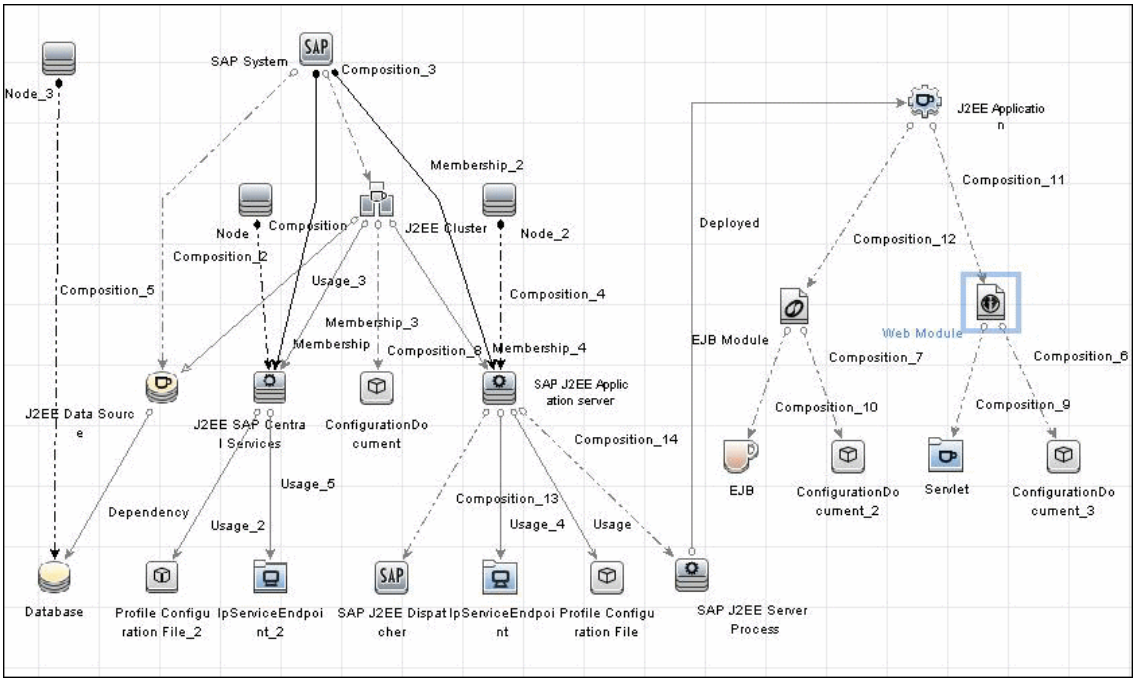
UCMDB discovers the SAP Application Server Java, which provides a Java 2 Enterprise Edition (Java EE) environment for developing and running Java EE programs.

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; 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 237.

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



How to Discover SAP Java

The SAP for Java discovery process enables you to discover SAP JAVA architecture and J2EE applications on the SAP JAVA server.

This task includes the following steps:

1. ## Prerequisite - Set up protocol credentials

   The SAP JMX protocol enables connection to a machine and verification whether an SAP system is installed on it.

   For credential information, see "Supported Protocols" on page 82.

2. ## Prerequisite - Add .jar files to Data Flow Probe machine

   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 the SAP system machine.

   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 the SAP system 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 6.4 and 7.0, in the **sap** folder, create two subfolders called **6.x** and **7.x,** and place the relevant .jar files into these folders.

3. ## Run the discovery

   In the Discovery Control Panel window, activate the jobs in the following order:

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

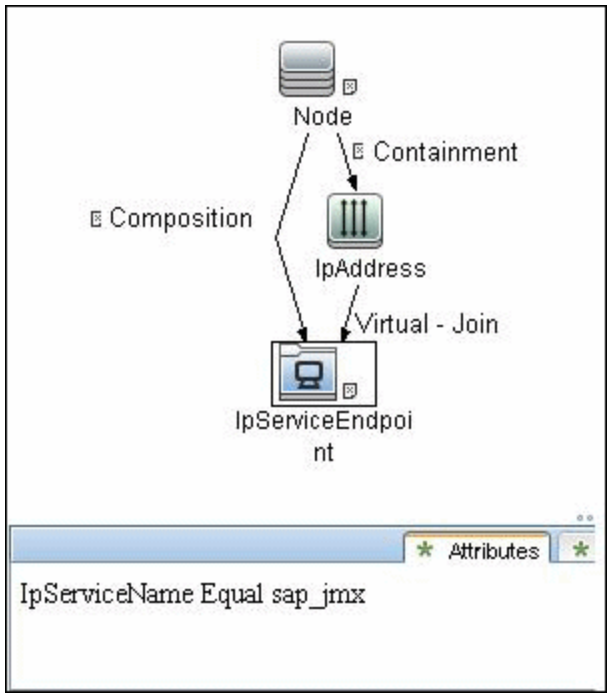
   - **Range IPs by ICMP**
   - **Host Connection By Shell**
   - **Host Resources and Applications by Shell**. Discovers SAP running software and processes.
   - **SAP TCP Ports**

- **SAP Java Topology by SAP JMX**. Discovers infrastructure entities in the SAP J2EE system: hosts, application servers, databases. Interfaces, Libraries, and Services are discovered as configuration files.

SAP Java Topology by SAP JMX Job

Trigger Query

- **Trigger CI:IpAddress**
- **Trigger query:**



Used Script

- **sap_j2ee.py**

Discovered CITs

- **Composition**
- **ConfigurationDocument**
- **Containment**
- **Database**
- **Dependency**
- **Deployed**
- **EJB**
- **EJB Module**

- **Entity Bean**
- **IpAddress**
- **IpServiceEndpoint**
- **J2EE Application**
- **J2EE Cluster**
- **J2EE Domain**
- **J2EE SAP Central Services**
- **JDBC Data Source**
- **Membership**
- **Message Driven Bean**
- **Node**
- **RunningSoftware**
- **SAP J2EE Application Server**
- **SAP J2EE Dispatcher**
- **SAP J2EE Server Process**
- **SAP System**
- **Servlet**
- **Stateful Session Bean**
- **Stateless Session Bean**
- **Usage**
- **Web Module**

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 13

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: **Discovery Control Panel** > 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.

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

   This discovery solution is based on the SAP protocol**.**

   For credential information, see "Supported Protocols" on page 82.

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 "Discovery Control Panel" 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 14

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

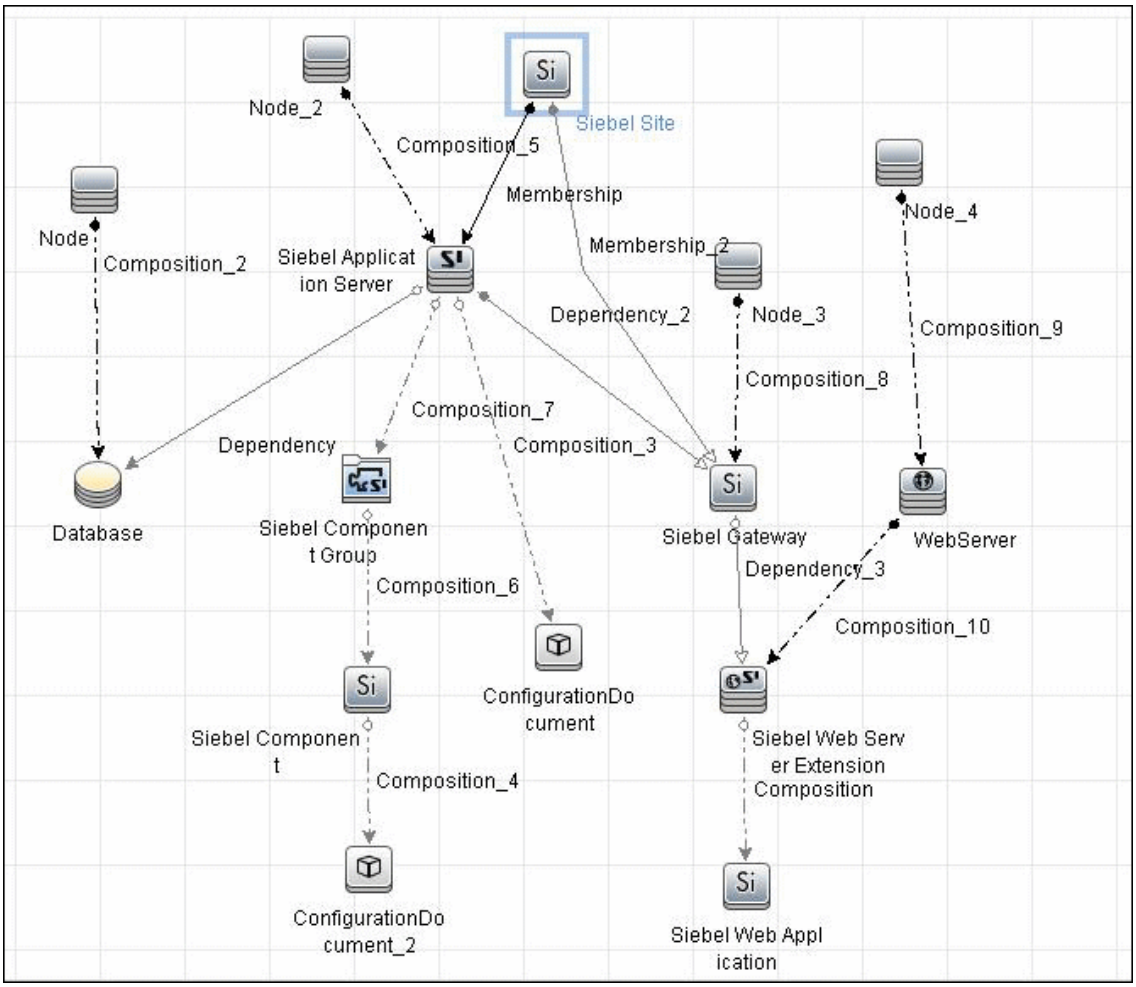
This discovery solution supports the following servers:

- Siebel 7.5
- Siebel 7.7
- Siebel 8.0
- Siebel 8.1

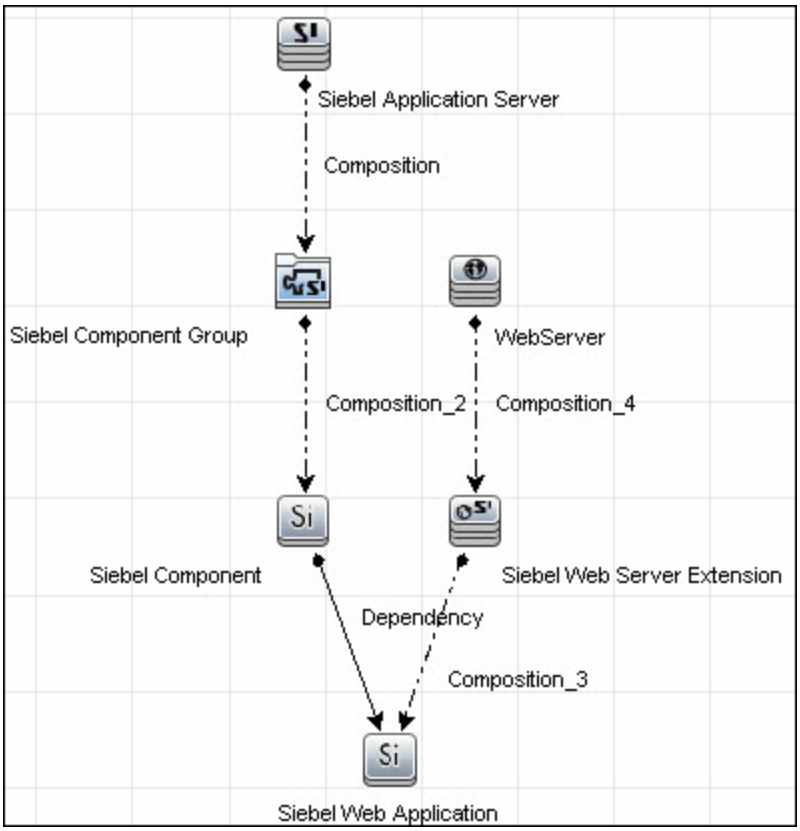
Topology

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 and includes the following steps:

- "Prerequisite - Set up protocol credentials" below
- "Prerequisites - Other" on next page
- "Run the discovery" on next page

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

   Set up the following protocols:

Platform	Protocol
Windows	▪ WMI protocol ▪ NTCMD protocol ▪ Siebel Gateway protocol
UNIX	▪ SSH protocol

Platform	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" on page 82.

2. **Prerequisites - Other**

The driver tool is used to extract data about the enterprise structure from Siebel.

- 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:

a. Copy the driver Command Line Interface (CLI) tool from the Siebel server to any folder on the Data Flow Probe machine.

b. (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.

c. 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.

3. **Run the discovery**

a. 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*.

b. In the Discovery Control Panel window, activate the jobs in the following order:

- **Class C IPs by ICMP** and **Host Connection by WMI**
- **Siebel DB by TTY**

c. Activate the following jobs to discover the Web tier:

- TCP Ports
- **Siebel Web Applications by NTCMD or UDA**, **Siebel Web Applications by TTY** and **Siebel DB by WMI and NTCMD**.
- WebServer Detection using TCP Ports.

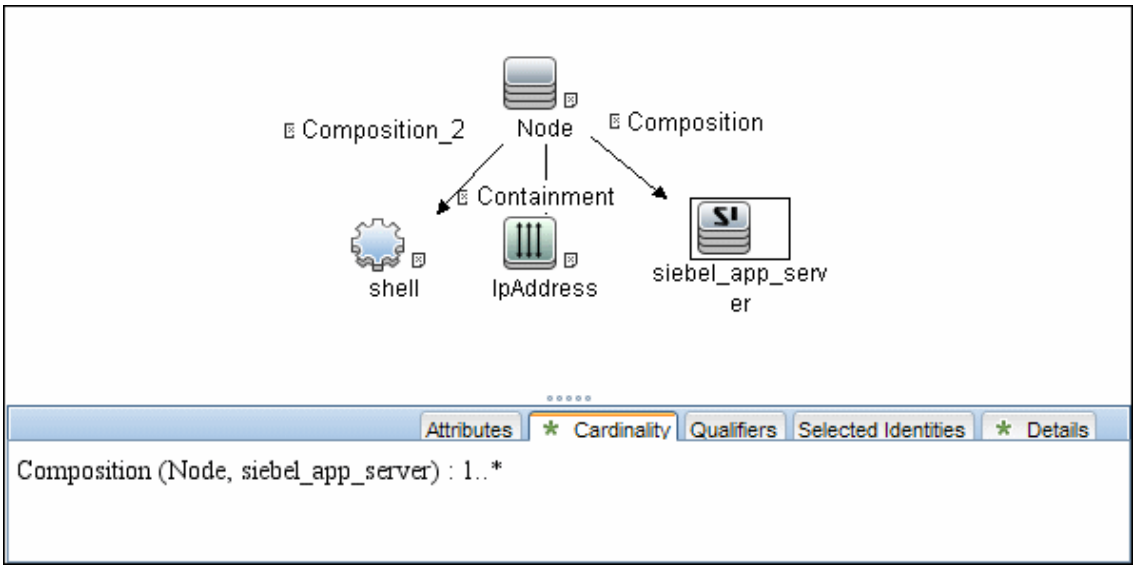
d. Activate all the jobs in the **Siebel** module to discover Siebel.

> **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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Siebel Application Server Configuration Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_APP_SERVER_CONFIG** adapter.

Used Scripts

- **file_ver_lib.py**
- **siebel_discover_appserver_config.py**

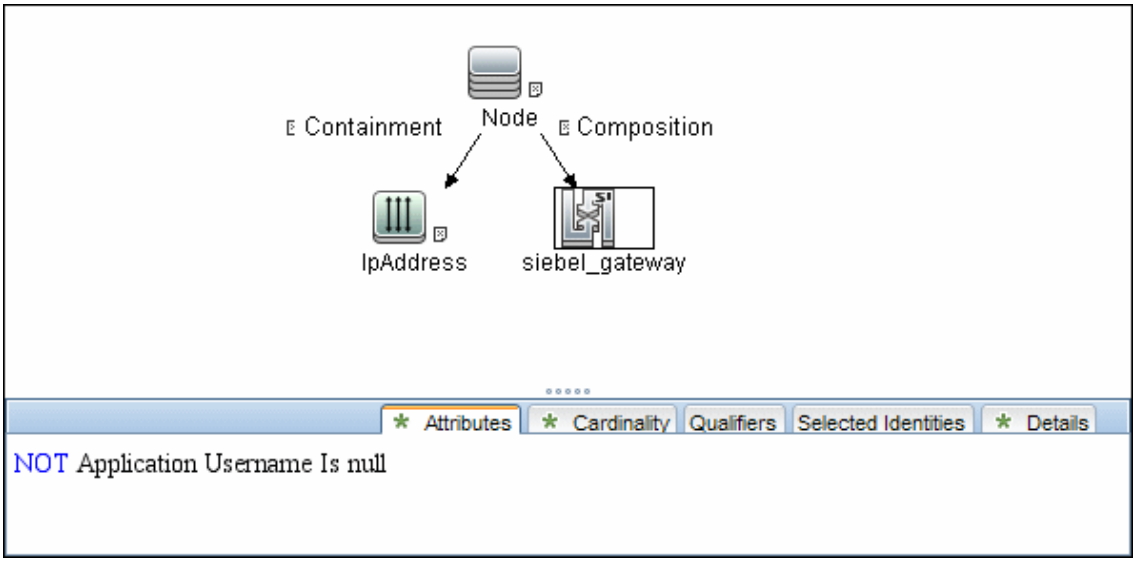
Discovered CITs

- **Composition**
- **ConfigurationDocument**
- **Siebel Application Server**

Note: To view the topology, see "Siebel Topology View" on page 242.

Siebel Application Servers Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_APP_SERVERS** adapter.

Used Scripts

- **siebel_common.py**
- **siebel_discover_enterprise.py**

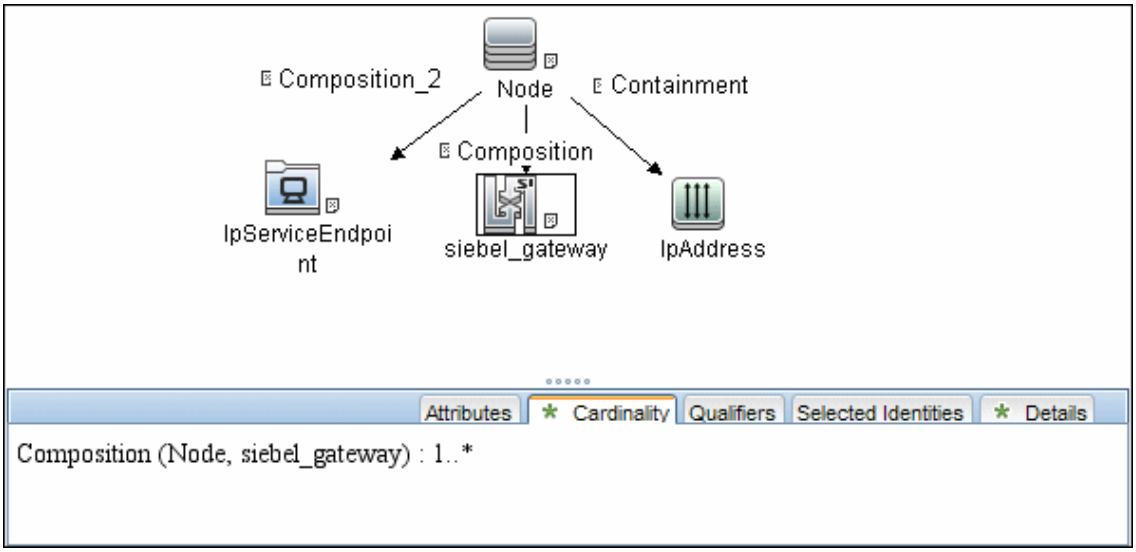
Discovered CITs

- **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 242.

Siebel Gateway Connection Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_GATEWAY_CONNECTION_(GTWY)** adapter.

Used Scripts

- **siebel_common.py**
- **siebel_discover_gateway.py**

Discovered CITs

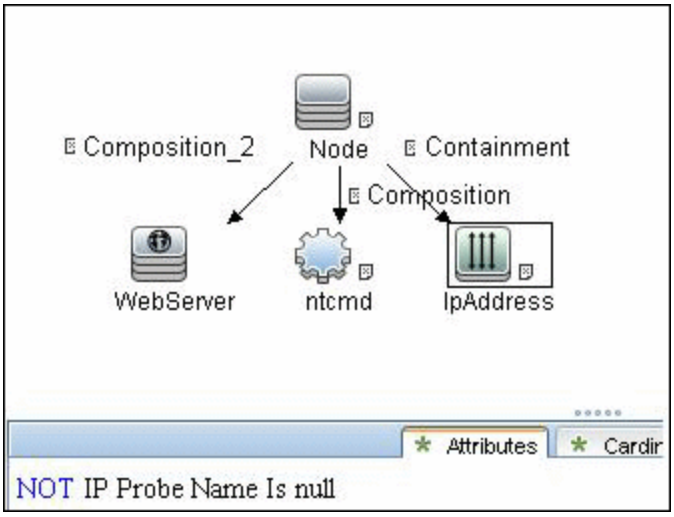
For details on the CIs that are discovered, see the Statistics table in the **Details** tab.

- **Composition**
- **Membership**
- **Siebel Enterprise**
- **Siebel Gateway**

Note: To view the topology, see "Siebel Topology View" on page 242.

Siebel Web Applications by NTCMD or UDA Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_WEBAPPS_NT** adapter.

Used Scripts

- **NTCMD_HR_REG_Software_Lib.py**
- **siebel_discover_wse.py**

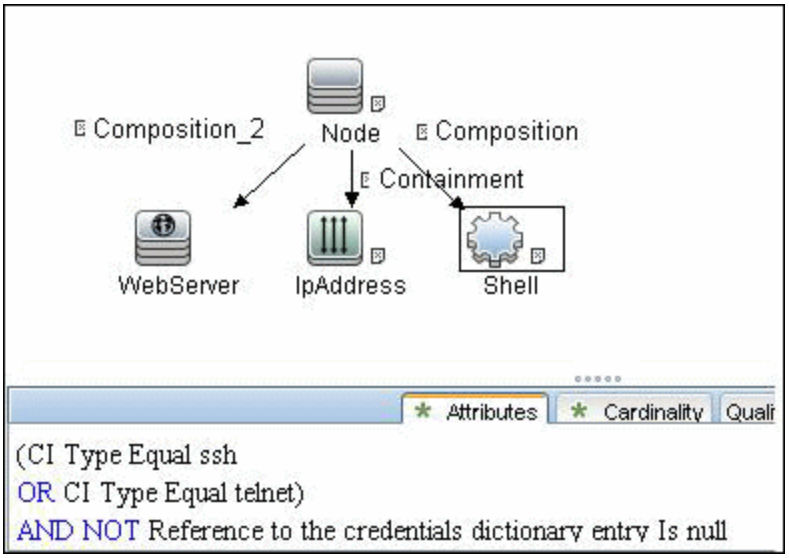
Discovered CITs

- **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 243.

Siebel Web Applications by TTY Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_WEBAPPS_UNIX** adapter.

Used Script

- **siebel_discover_wse.py**

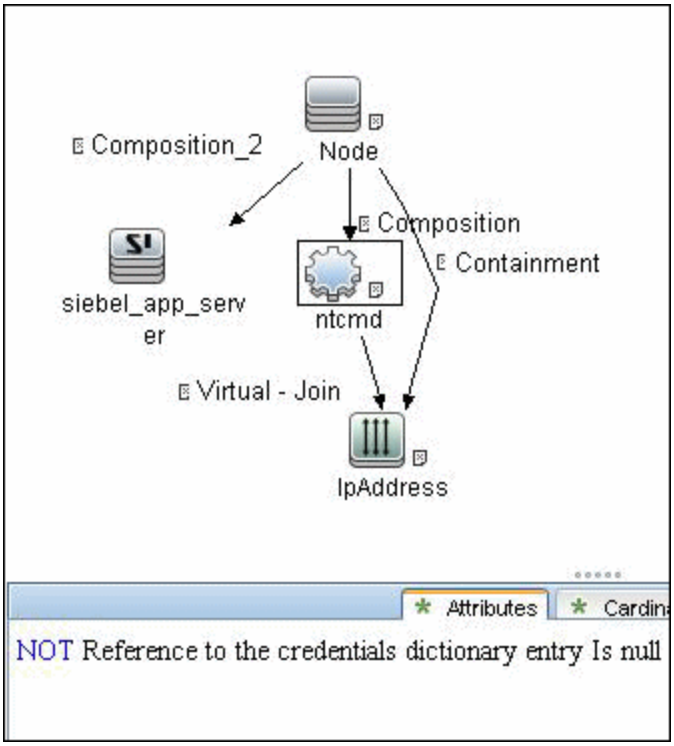
Discovered CITs

- **Composition**
- **Configuration Document**
- **Containment**
- **Dependency**
- **IpAddress**
- **Node**
- **Route**
- **Siebel Enterprise**
- **Siebel Gateway**
- **Siebel Application**
- **Siebel Web Server Extension**
- **WebServer**

Note: To view the topology, see "Siebel Web Topology View" on page 243.

Siebel DB by NTCMD or UDA Job

Trigger Query



Adapter

This job uses the **SIEBEL_DIS_DB_NT** adapter.

Used Script

- **siebel_discover_odbc.py**

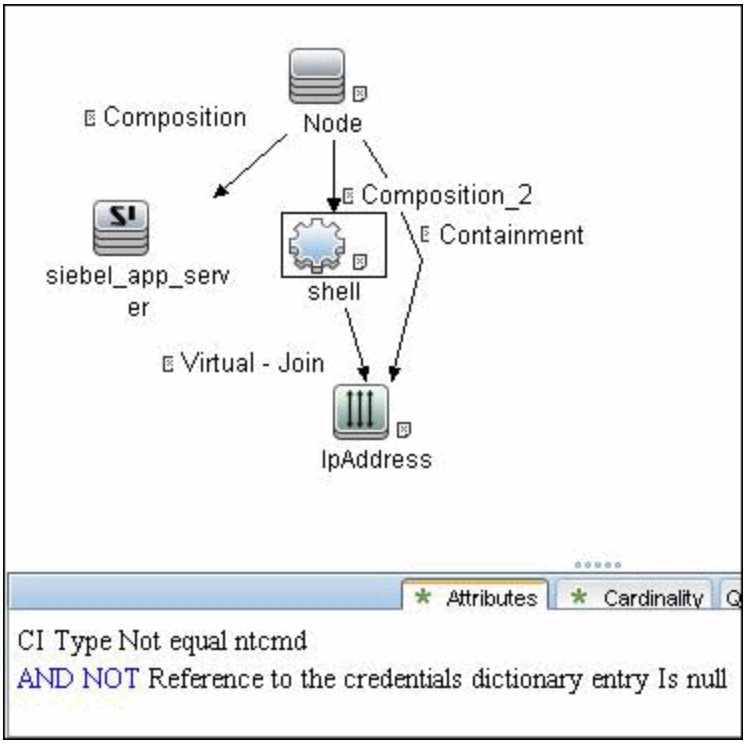
Discovered CITs

- **Composition**
- **Containment**
- **Database**
- **Dependency**
- **IpAddress**
- **Node**

Note: To view the topology, see "Siebel Topology View" on page 242.

Siebel DB by TTY Job

Trigger Query



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 242.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for Siebel discovery.

- 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 15

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

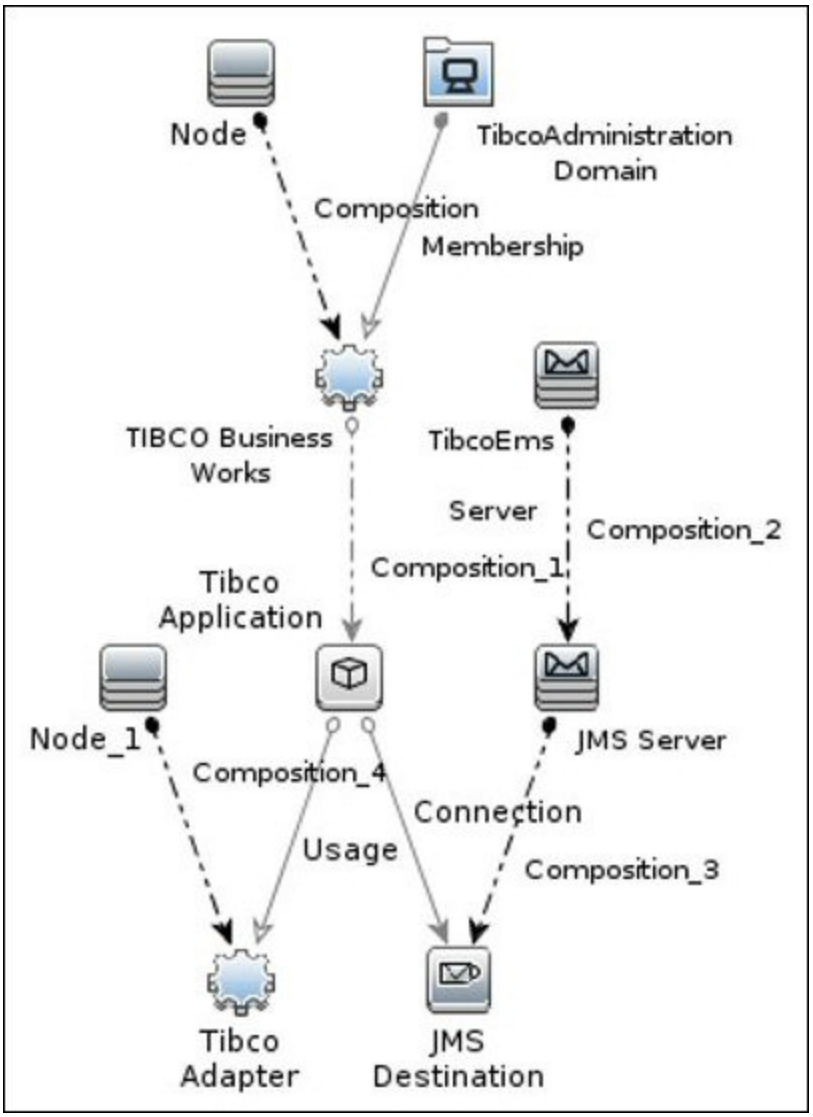
TIBCO discovery supports the following versions of software running in a UNIX environment:

- 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 "Discovered CITs" on page 260.



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" on page 82.

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 Resources and 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

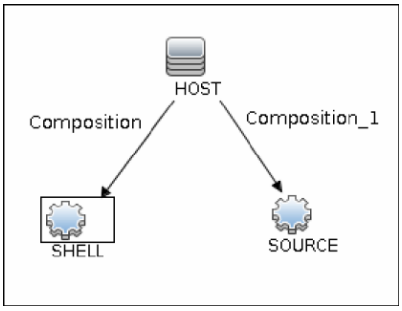
This section gives details about the TIBCO BusinessWorks by Shell job.

Input CIT

TibcoBusinessWorks

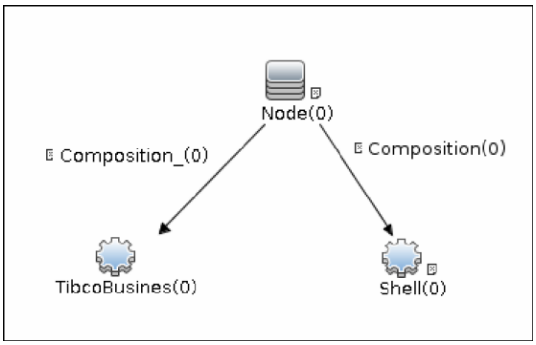
Input TQL Query

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



Trigger TQL Query

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



Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}
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

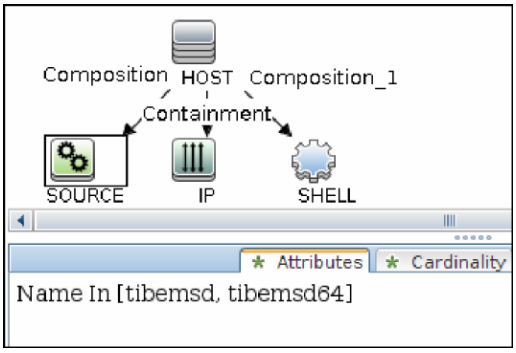
This section gives details about the TIBCO EMS by Shell job.

Input CIT

Process

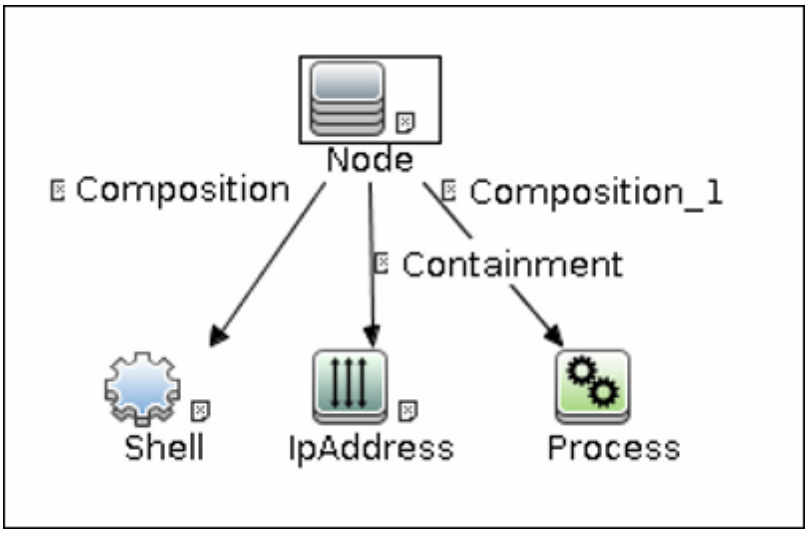
Input TQL Query

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}
credentialsId	${SHELL.credentials_id}
hostId	${HOST.root_id}
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 16

UDDI Registry Discovery

This chapter includes:

Overview

The UDDI discovery process enables you to discover Web services from a UDDI registry.

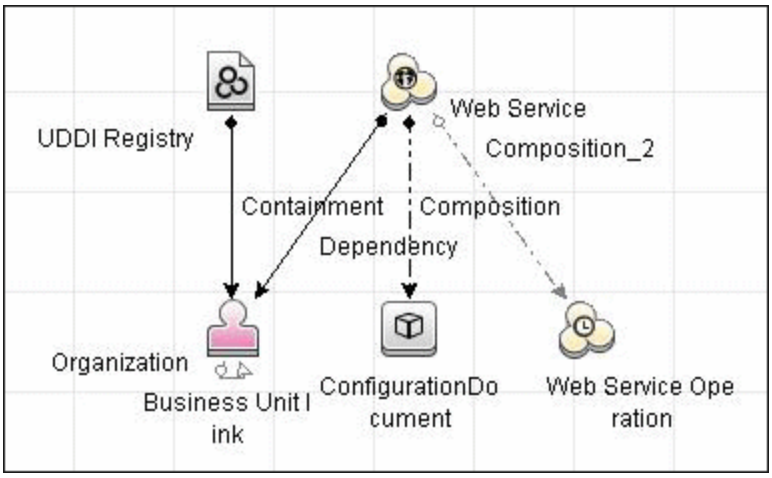
DFM queries the UDDI registry for its Web services, including non-SOAP services, or for a specific publisher service (if defined in the UDDI Registry protocol). The Web services found in the UDDI registry are represented by a **WebService Resource** CI in the CMDB and the registry is created as a **UDDI Registry** CI.

Supported Versions

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. **Prerequisite - Set up protocol credentials**

   Set up the UDDI protocol.

   For credential information, see "Supported Protocols" on page 82.

2. **Run the discovery**

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

   Activate the following jobs:

- **WebServices by URL**
- **Webservice Connections by UDDI Registry**
- **Webservices by UDDI Registry**

3. ## 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*.

Chapter 17

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.

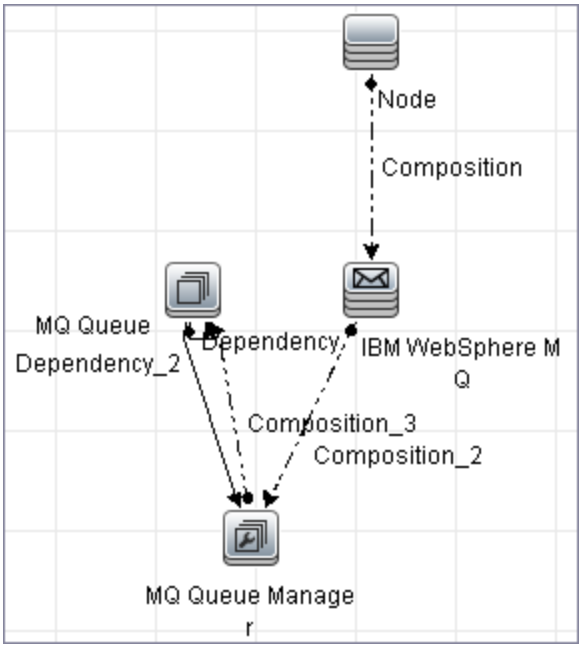
Note:

- 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 279.

This section describes the following views:

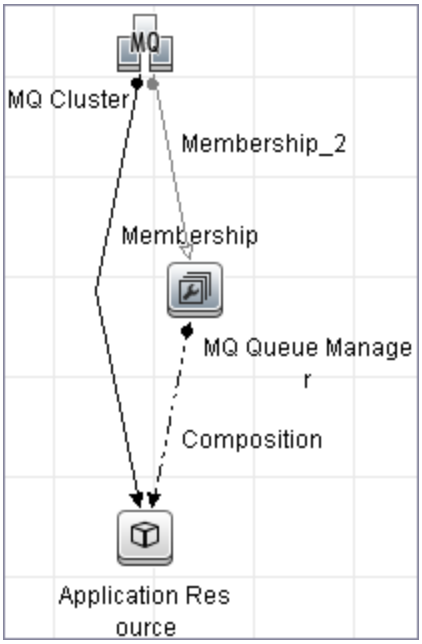
MQ Queue Dependency

This view displays queues that are dependent on other MQ objects and typically include Remote Queues, Alias Queues, and Remote Queue Managers:



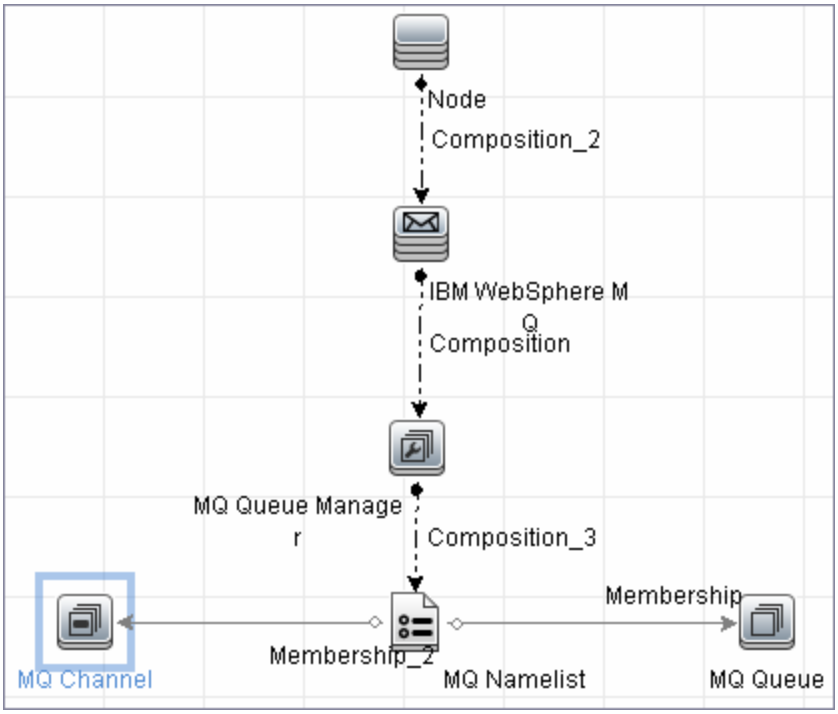
MQ Q Manager Resources on Non-Local Cluster

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.



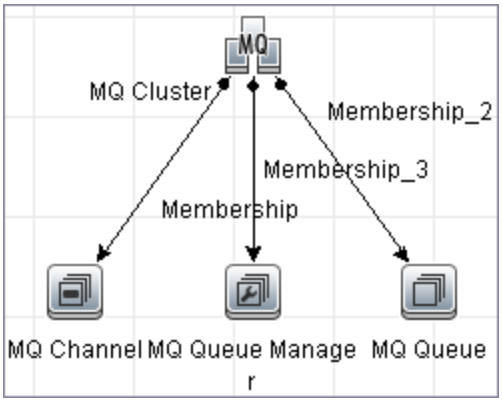
MQ Namelist Membership

This view displays namelists and their members:



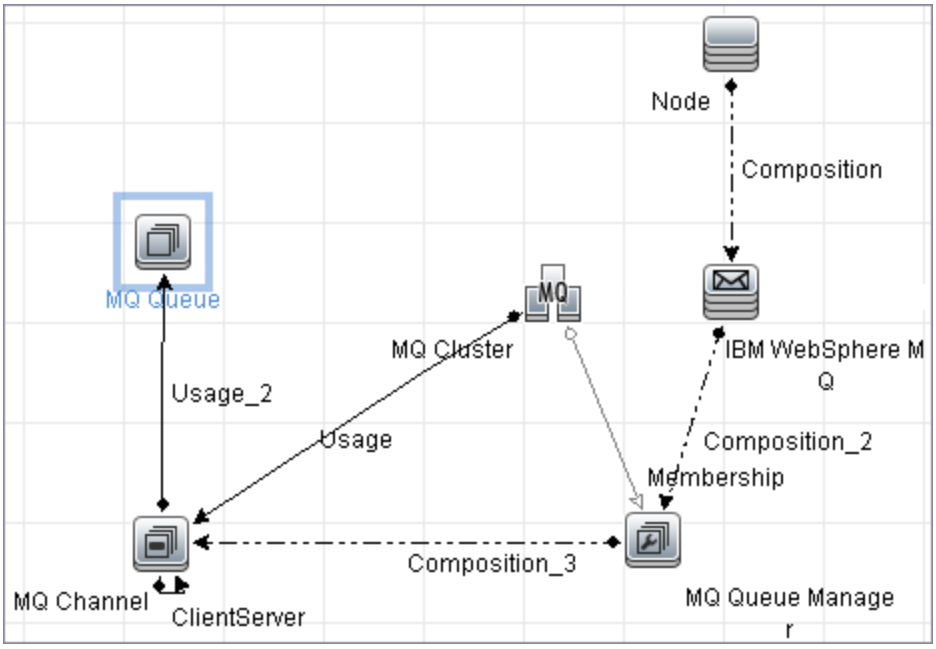
MQ Cluster Membership

This view displays clusters and their members:



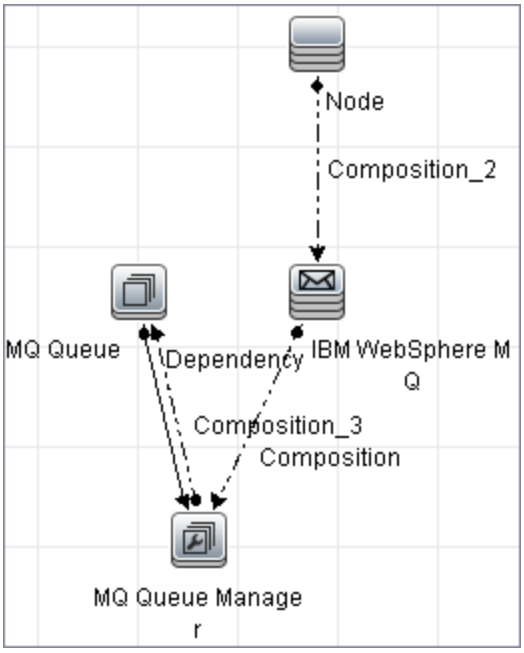
MQ Channel Communication

This view displays client-server communication between MQ Channels and queues used by the channels:



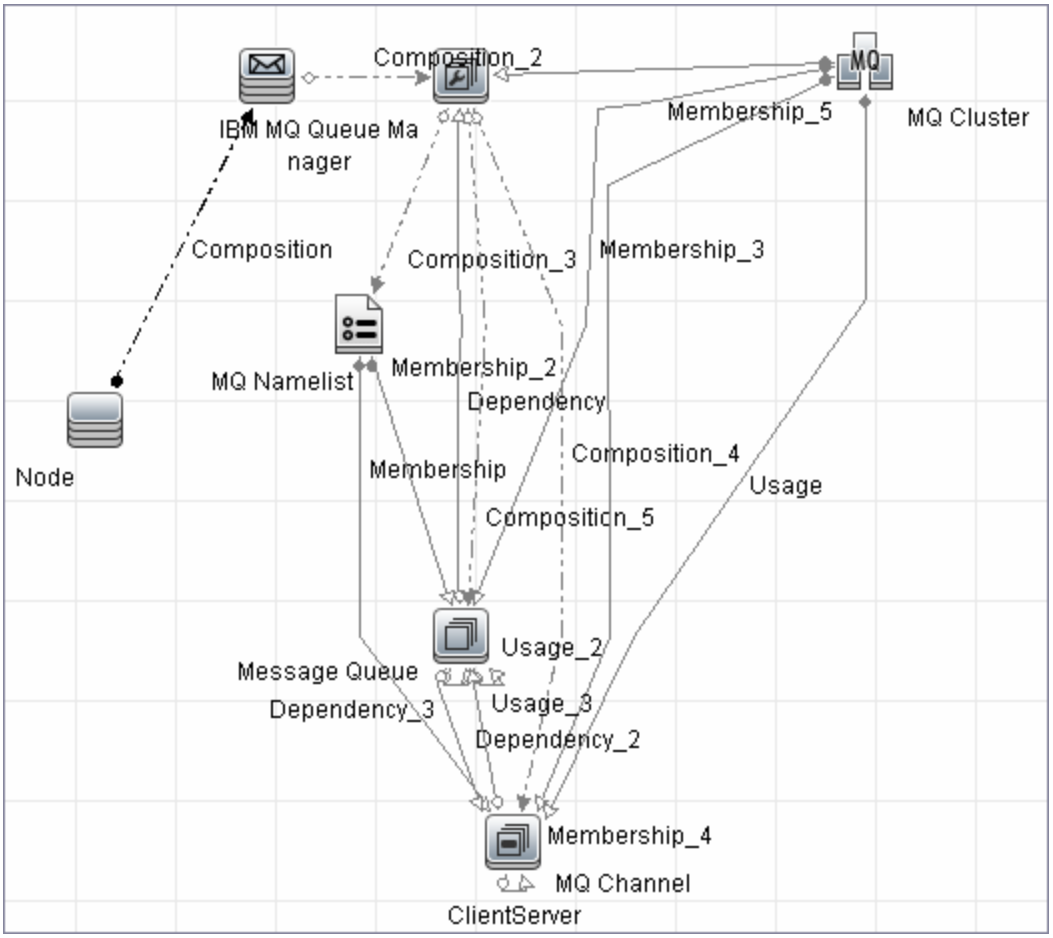
MQ Alias Queue Managers

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:



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" on page 82.

   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 "Add/Edit IP Range Dialog Box" 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 "Details Pane (Protocol)" 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 Resources and 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

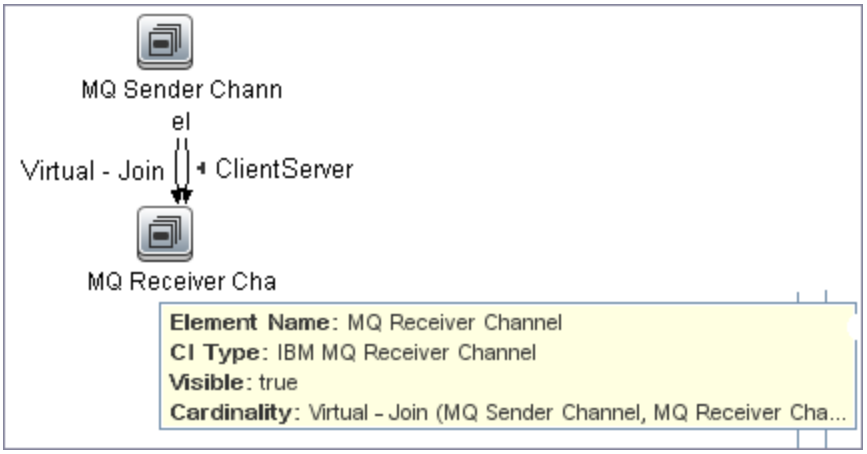
This discovery uses the **WebSphere MQ Topology by shell** adapter.

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.



Discovered CITs

The WebSphere MQ discovery discovers the following CI Types. For details on viewing the discovered CITs,, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Note: To view the topology, see "Topology" on page 269.

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.
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.

CI Type	Key Attributes	Description
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.
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.
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

Link	End1	End2	Cardinality	Description
				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.
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 279.
- If the discovery results appear incomplete, try increasing the value of the **mq_cmd_timeout** parameter. For details, see " Adapter Parameters" on page 279.

Part III: Clusters

Chapter 18

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.

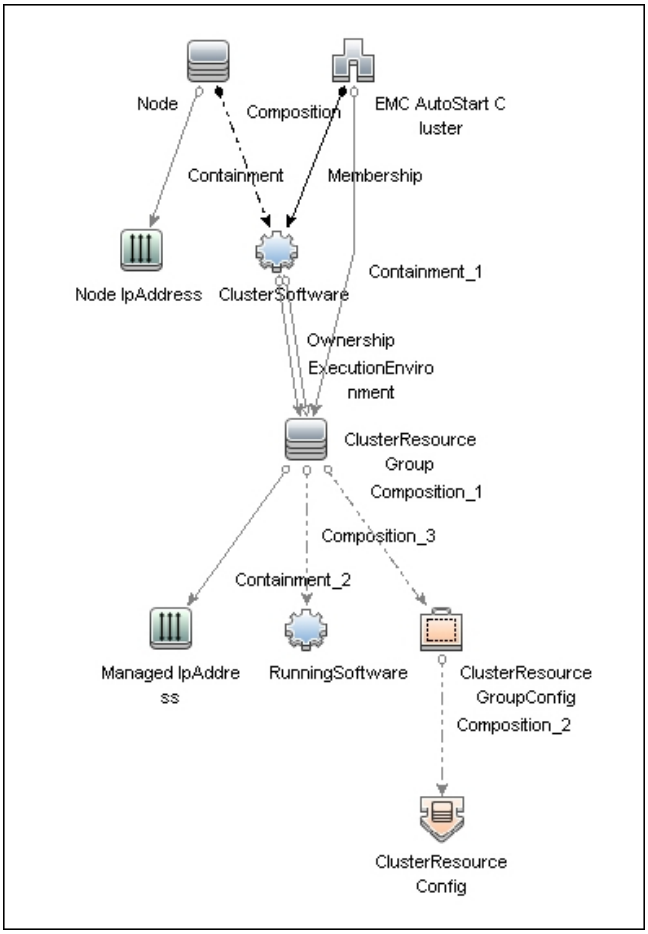
Supported Versions

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 291.



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" on page 82.

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 294.

   c. In Windows, if connecting with **NTCMD**, run the **xCmdSvc** 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 Resources and 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

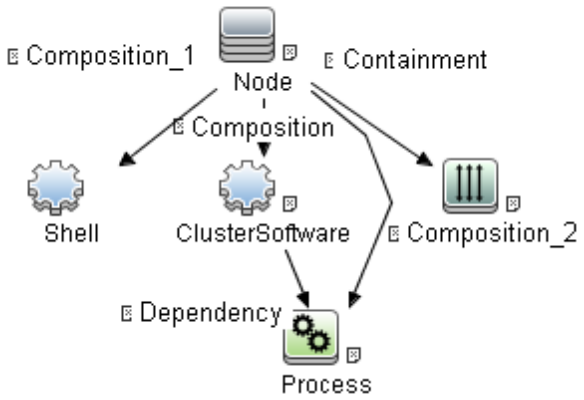
This section gives details about the EMC AutoStart by Shell job.

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.

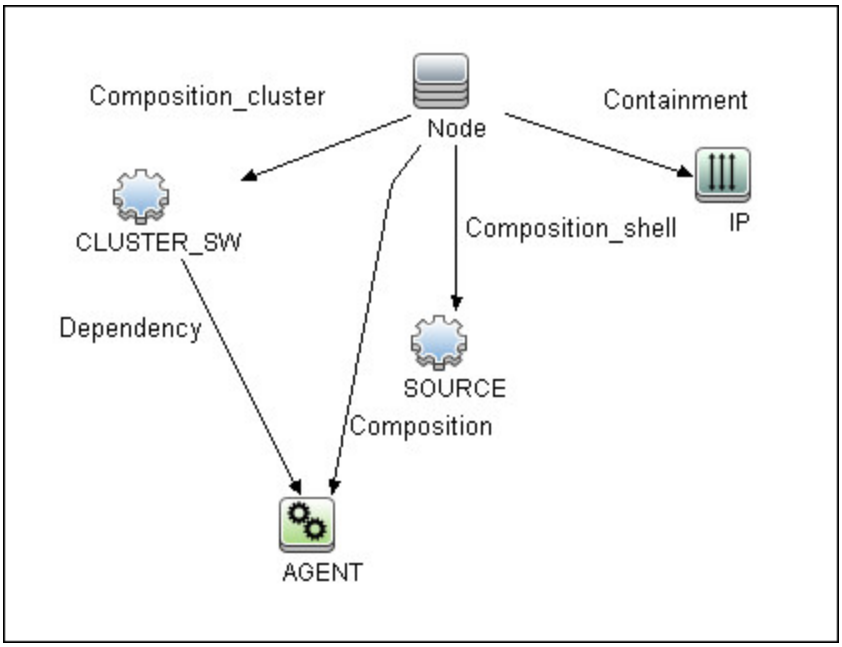
EMC_AutoStart_by_Shell Adapter

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}

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**

Global Configuration Files

None

Parameters

None

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 Signature** 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 19

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.

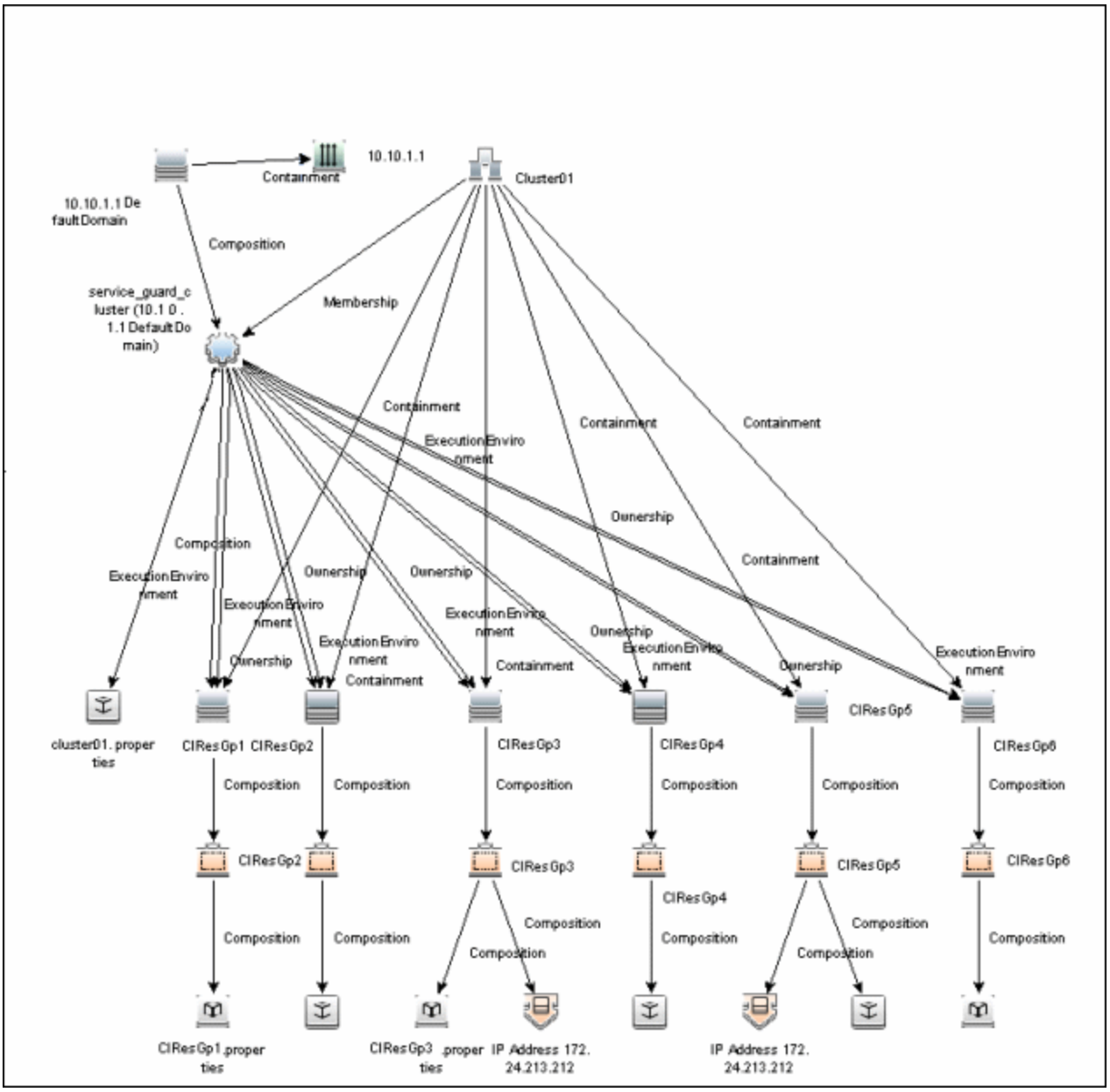
Supported Versions

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 "Discovered CITs" on page 302.



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" on page 82.

3. **Run the discovery**

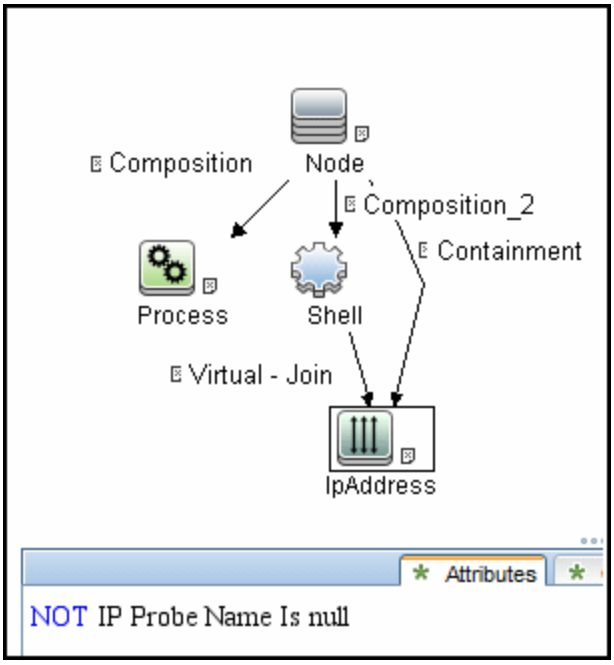
   Run the following jobs:

   - **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 Resources and 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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Service Guard Cluster Topology by TTY Job

Trigger Query



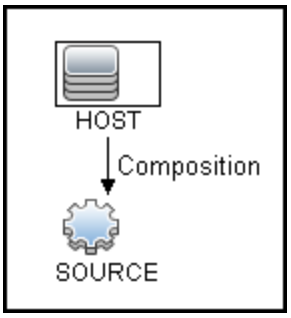
Adapter

This job uses the **Service Guard Cluster Topology** adapter.

For details, see "Service Guard Cluster Topology Adapter" on next page

Service Guard Cluster Topology Adapter

Input Query



Used Scripts

- Service_Guard_Cluster_Topology.py
- file_mon_utils.py
- file_ver_lib.py
- service_guard_discoverers.py
- file_system.py
- file_topology.py
- service_guard_topology.py
- service_guard.py
- oracle_shell_utils.py

Created/Changed Entities

Entity Name	Type	Description
Service_Guard_Cluster_Topology.py	Script	Discovery script
service_guard_discovers.py	Script	Discovery script
service_guard_topology.py	Script	Discovery script
service_guard.py	Script	Discovery script
oracle_shell_utils.py	Script	Discovery script

Discovered CITs

- Clustered Software
- Composition
- ConfigurationDocument
- Containment
- Dependency
- ExecutionEnvironment
- ipAddress
- Membership
- Node
- Ownership
- SG Package
- SG Resource
- Service Guard Cluster
- RunningSoftware

Note: To view the topology, see "Topology" on page 298.

HP Serviceguard Cluster Commands

This section includes the Serviceguard clustering commands.

Command
/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

Command
`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.

Command
`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.

Command
`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.

Command
`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.

Command
`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.

Command
`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.

Command
`/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.

Command
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.

Command
`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 20

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 And Applications by Shell** job.

   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.

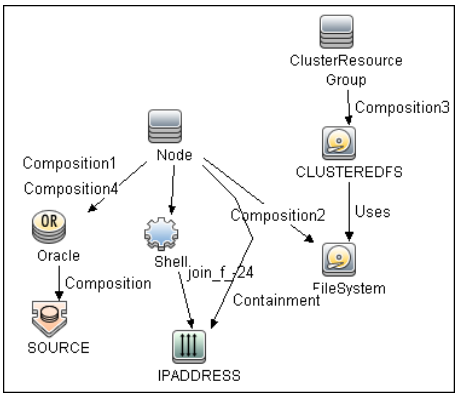
      For details on running jobs, refer to "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide.*

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

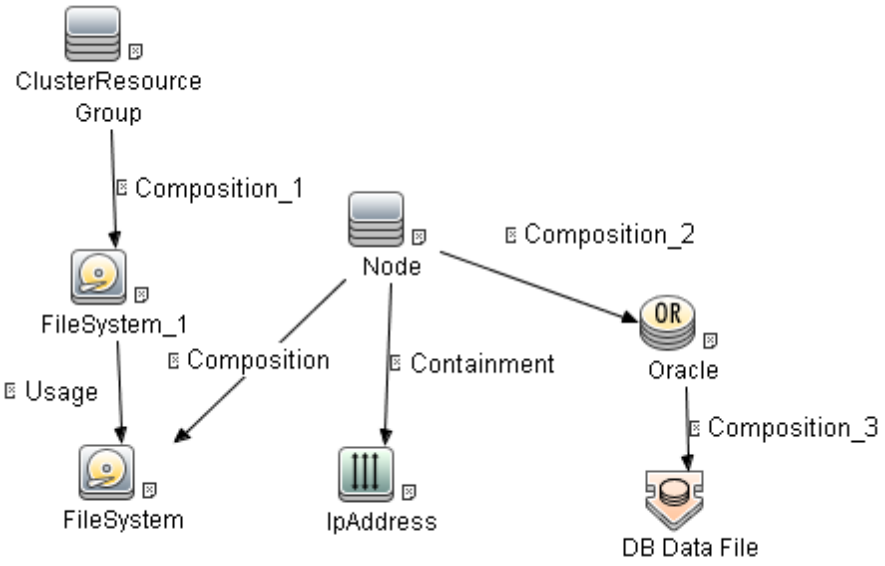
- **linkDbDatafileAndFs.py**

Discovered CITs

- **DB Data File**
- **FileSystem**
- **Node**
- **Usage**

The Link DB DataFiles and Clustered FS 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 which 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 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 which 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.

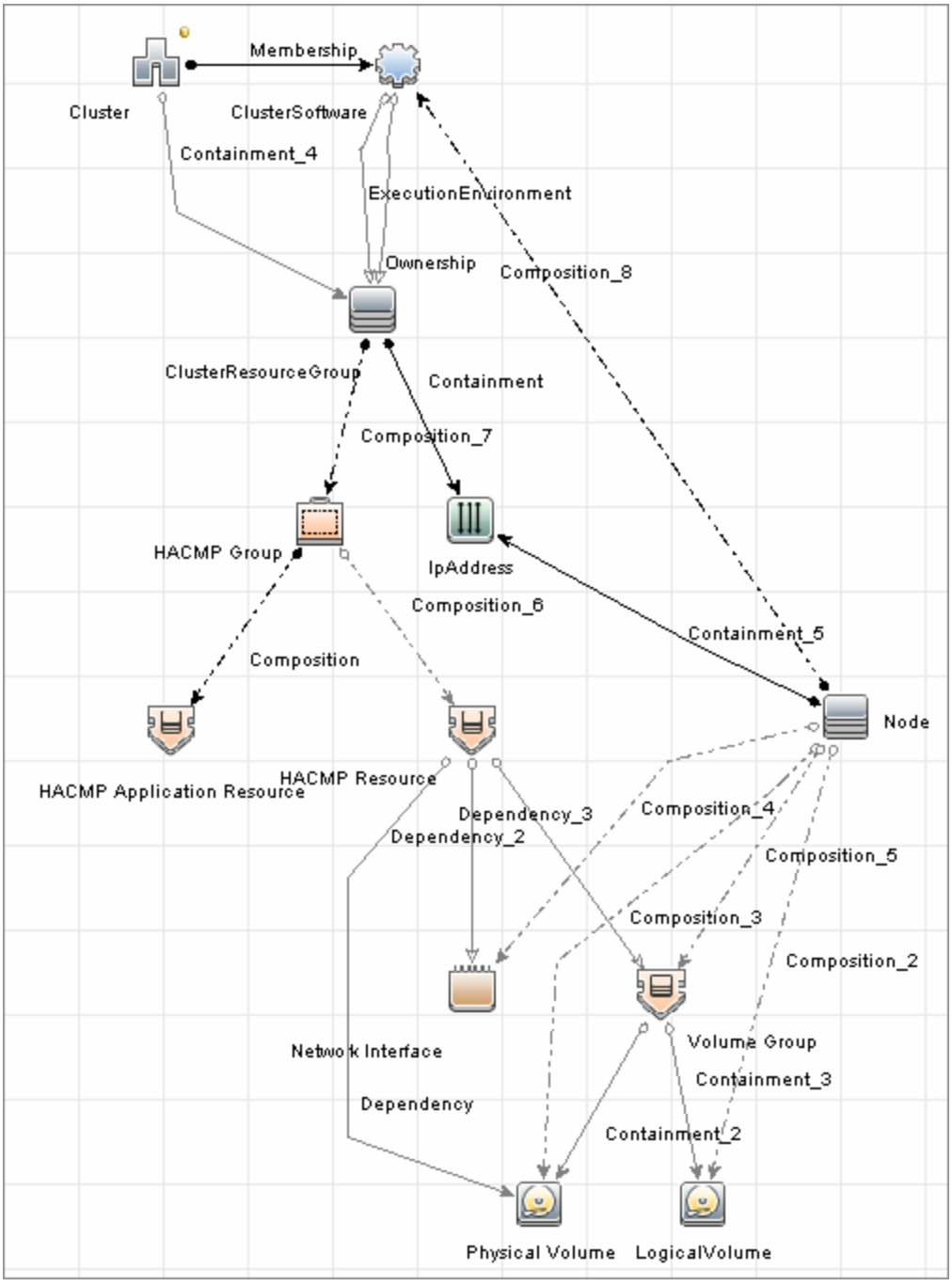
The **IBM_HACMP** package discovers HACMP on AIX via TTY (SSH or Telnet protocols). The package follows the discovery model to discover the HACMP Topology (configured networks, node interfaces-both public TCP/IP and serial heartbeat, and service IPs) and Application Resources (configured resource groups, 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" on page 82.

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 906.

   - 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.

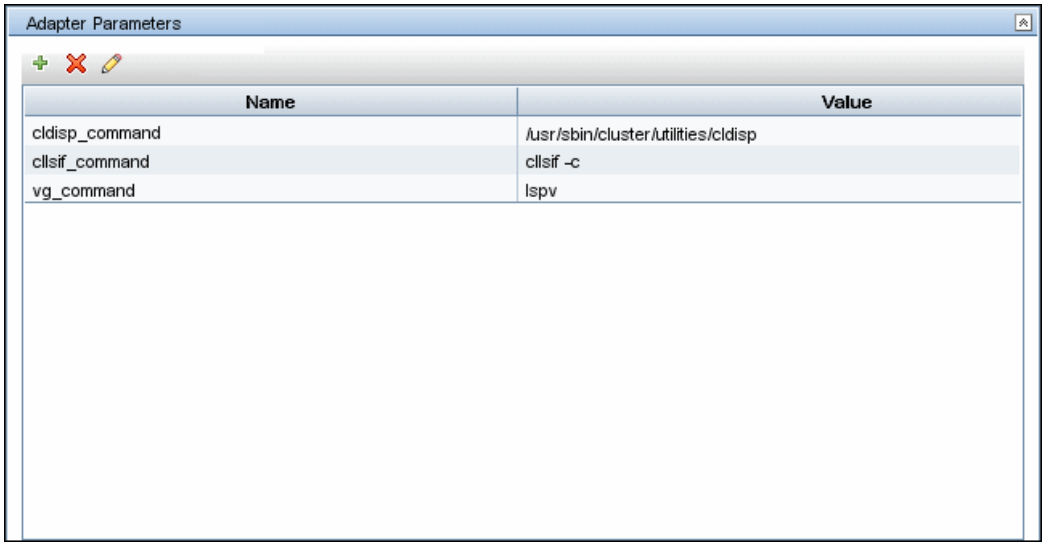
   c. Run the **Range IPs by ICMP** job to discover which of the machines in the IP range are up.

   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



HACMP Topology discovery adapters

e. Activate the **HACMP Topology Discovery** job. After the job completes, verify the creation of **HACMP** CIs through the Statistics Results pane. For details, see "Statistics Results Pane" in the *HP Universal CMDB Data Flow Management Guide*.

f. Activate the **HACMP Application Discovery** job. This job creates HACMP application and resource CIs.

For details on running jobs, refer to "Discovery Control Panel" 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" on next page
- "Get Cluster Information" on page 319
- "Get DNS Information from the Host File" on page 320
- "Get Volume Group Information" on page 321
- "Get HACMP Application Information" on page 322

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:

Example of output (*cont'd*)	```## 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 ## ############# ## ... ## ================```

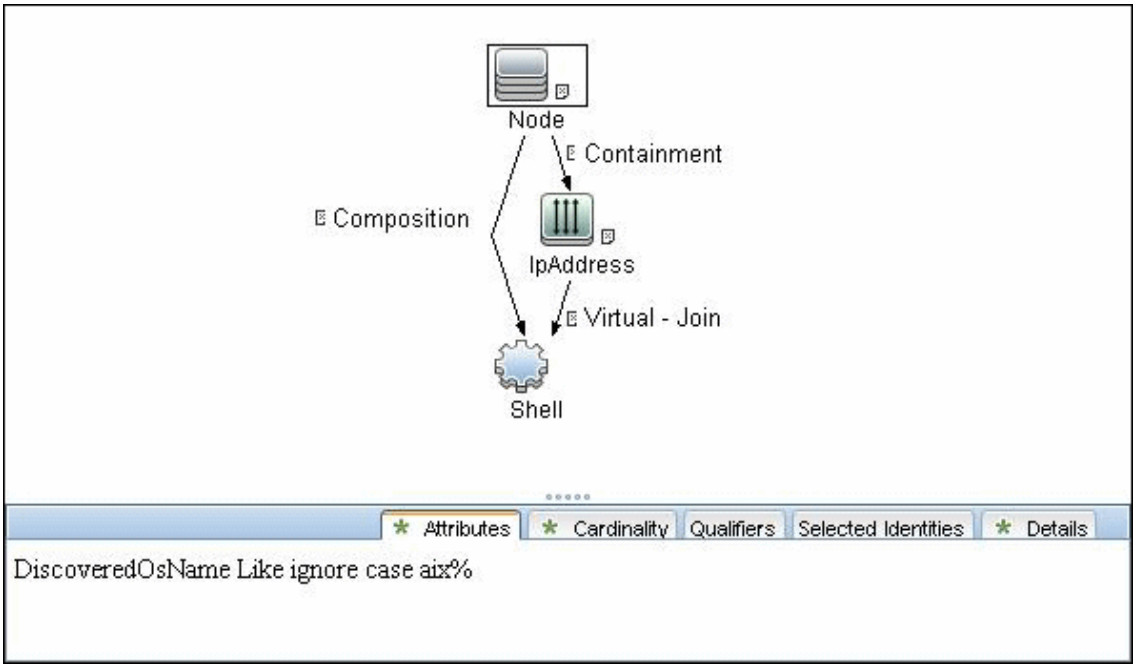
Values taken	Application information
Comments	This command retrieves the HACMP Application information.

HACMP Topology Discovery Job

This section describes the following:

Trigger Query (Shell not NTCMD HACMP)

This trigger requires a TTY Shell that is not an NTCMD Shell.



Adapter

- **Created/Changed Entities**
  - **Hacmpcluster CIT**
  - **Failoverclustersoftware CIT**
  - **Logical Volume**
  - **Physical Volume**
  - **Volume Group**
  - **Network Interface**

Used Script

- **TTY_HACMP_Applications.py**

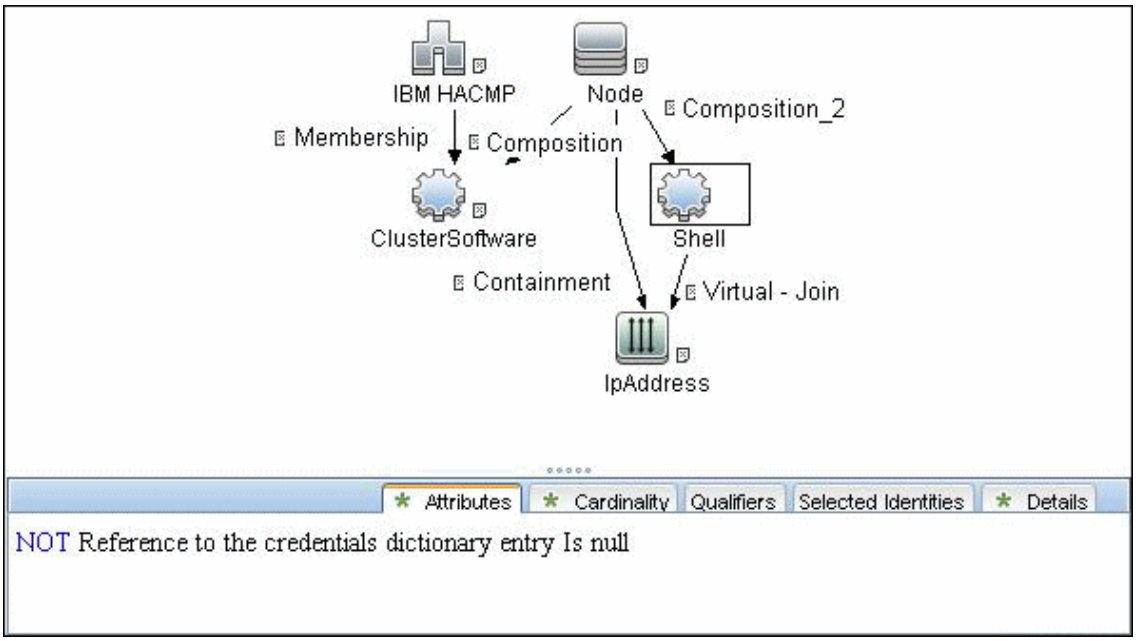
Discovered CITs

- **ClusterResourceGroup**
- **ClusterSoftware**
- **Composition**
- **Containment**
- **Depencency**
- **ExecutionEnvironment**
- **HACMP Cluster**
- **HACMP Resource**
- **HACMP Resource Group**
- **Interface**
- **IpAddress**
- **Membership**
- **Node**
- **Ownership**
- **Physical Volume**
- **RunningSoftware**
- **Usage**
- **Volume Group**

HACMP Application Discovery Job

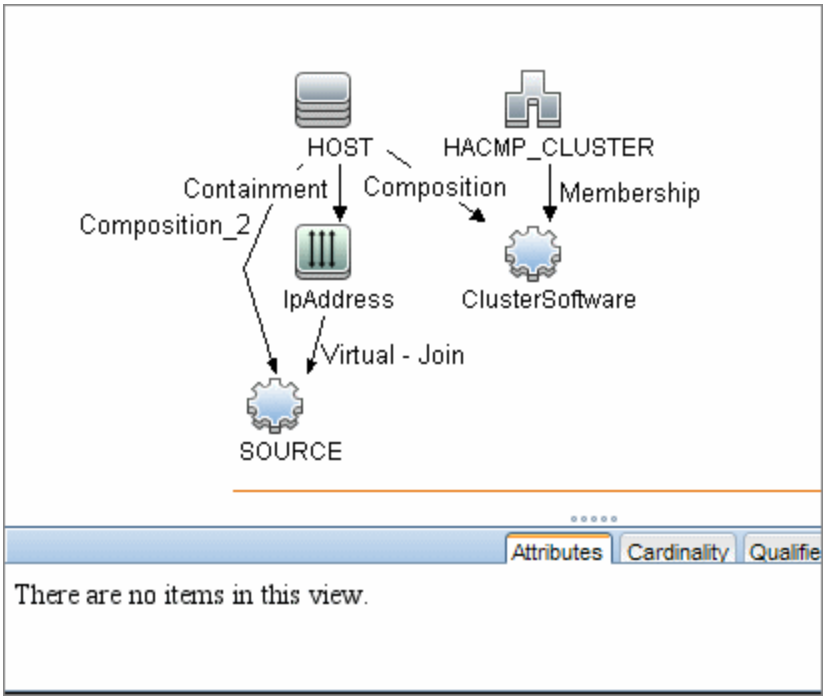
This section describes the following:

Trigger Query (Shell in HACMP Cluster)



Adapter

- **Input Query**



- **Created/Changed Entities**
  - **Hacmpgroup**
  - **Hacmpresource**
  - **Network Interface**
  - **Cluster Server**
  - **IpAddress**
  - **Physical Disk**
  - **Volume Group**

Used Script

- **TTY_HACMP_Topology.py**

Discovered CITs

- **ClusterSoftware**
- **Composition**
- **Containment**
- **HACMP Cluster**

- **Interface**
- **IpAddress**
- **LogicalVolume**
- **Membership**
- **Node**
- **Physical Volume**
- **Volume Group**

Chapter 22

Load Balancer Discovery

This chapter includes:

Overview

DFM discovers the following load balancers:

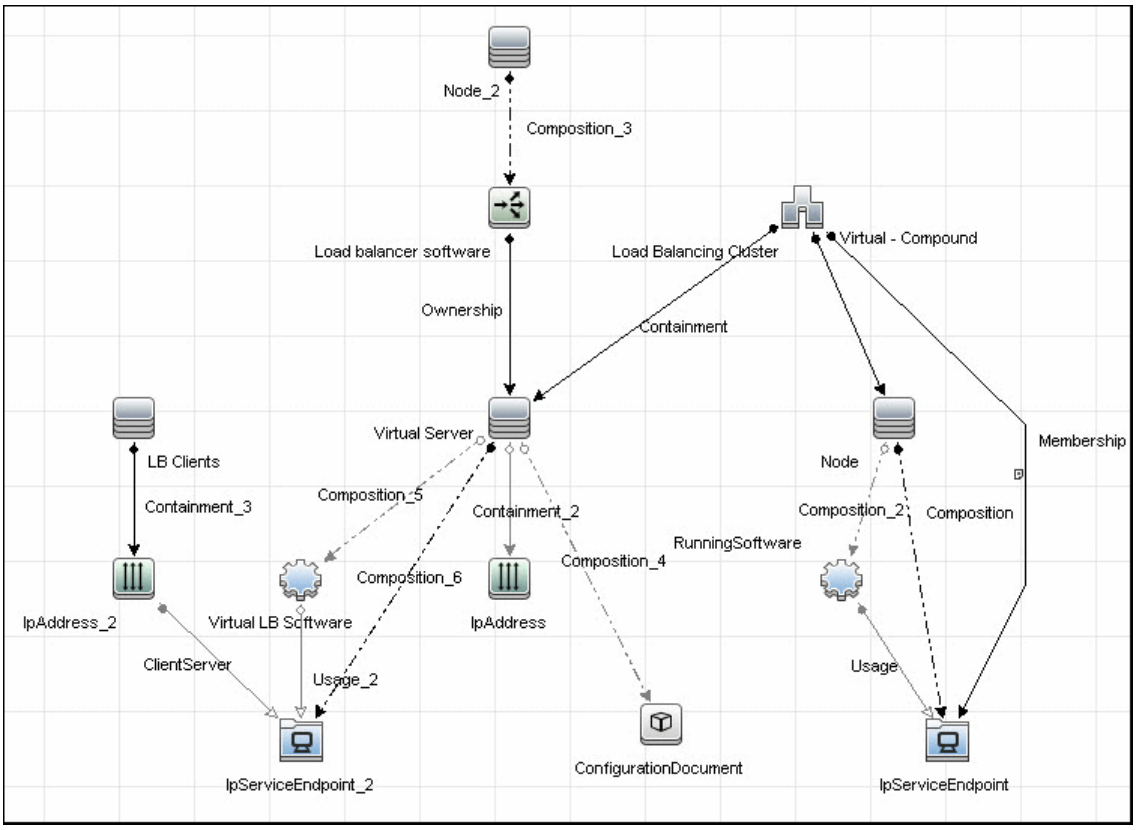
- F5 BIG-IP Local Traffic Manager (LTM)
- Nortel Application Switches (formerly known as Alteon Application Switches)
- Cisco Content Services Switches (CSS)

Supported Versions

The supported version for each load balancer is as follows:

- **F5 BIG-IP Local Traffic Manager:** versions 9 and 4
- **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 337.

How to Discover Load Balancers

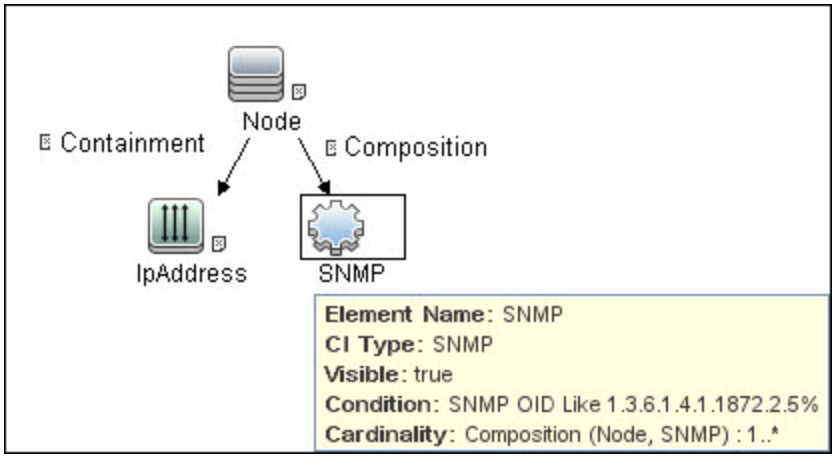
This task explains how to discover load balancers and includes the following steps:

- "Prerequisites" below
- "Run the discovery" on next page

1. ## Prerequisites

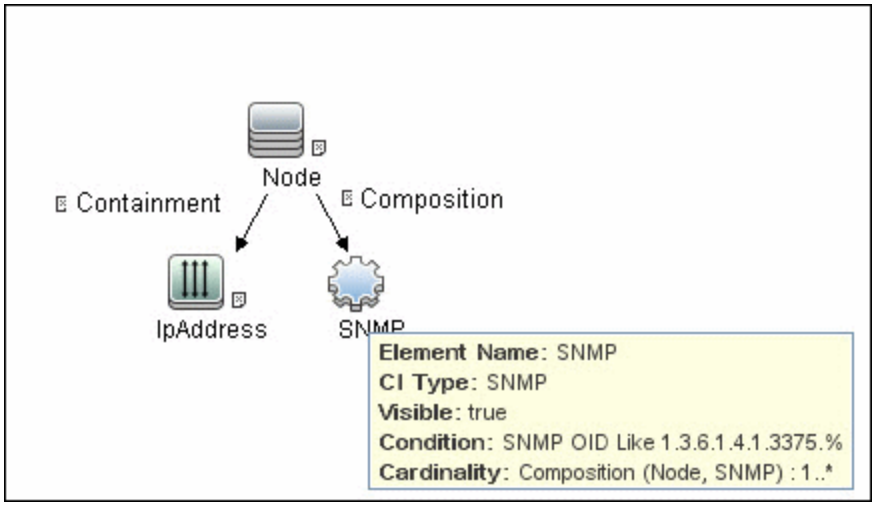
   Run the **Host Connection by SNMP** job to discover and create SNMP CIs which answer the following requirements:

   - To be the trigger query for the **Alteon application switch by SNMP** job with the following condition:



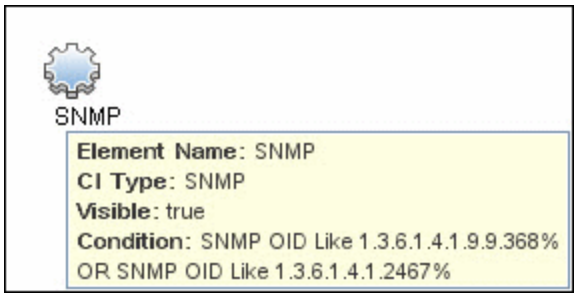

```
SNMP OID Like 1.3.6.1.4.1.1872.2.5%
```

- To be the trigger query for the **F5 BIG-IP LTM by SNMP** job with the following condition:




```
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:




```
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" on page 82.

2. **Run the discovery**

- **Host Connection by SNMP.** For details on the prerequisites to running a load balancer job, see "Prerequisites" on page 332.
- 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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Alteon_application_switch Job

This package contains a class model definition, an adapter, and a job used to discover Nortel application switches by SNMP.

To run this package, activate the **Alteon application switch by SNMP** job. DFM discovers Nortel (Alteon) load balancers and all related CIs.

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
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

F5_BIGIP_LTM Job

This package contains a class model definition, an adapter, and a job used to discover the F5 BIG-IP Local Traffic Manager (LTM) by SNMP. The package supports F5 BIG-IP LTM, versions 4 and 9.

To run this package, activate the **F5 BIG-IP LTM by SNMP** job. DFM chooses all SNMPs related to F5 and runs against them.

The following SNMP tables are queried for version 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
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

Cisco_CSS Job

This package contains a class model definition, an adapter, and a job used to discover Cisco Content Services Switches by SNMP. This package supports all versions of Cisco CSS.

To run this package, 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.

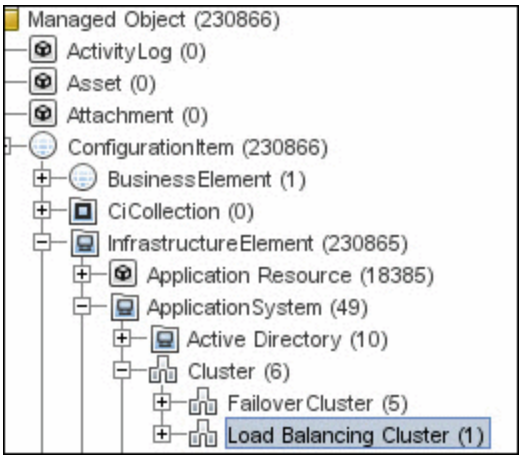
Table name	Name from MIB	OID
CNT	apCntTable	1.3.6.1.4.1.2467.1.16.4.1 or 1.3.6.1.4.1.9.9.3681.16.4.1
SVC	apSvcTable	1.3.6.1.4.1.2467.1.15.2.1 or 1.3.6.1.4.1.9.9.3681.15.2.1
CNT to SVC	apCntsvcEntry	1.3.6.1.4.1.2467.1.18.2.1 or 1.3.6.1.4.1.9.9.3681.18.2.1

Discovered CITs

The following CITs model load balancer topology:

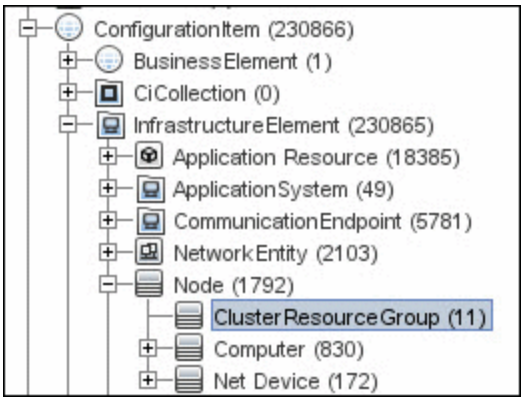
- **Load Balancer Software**

  This CIT represents software that provides load balancing solutions. For details on the supported load balancers, see "Overview" on page 331.



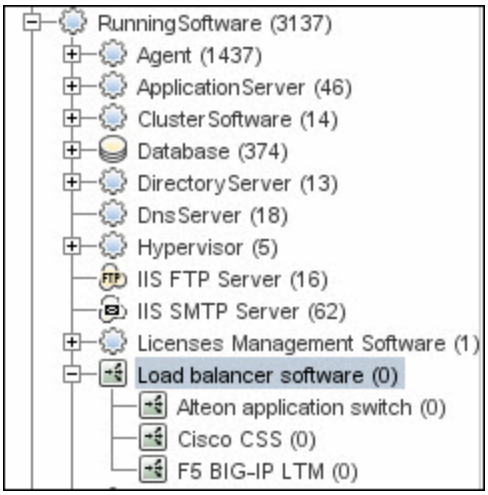
- **Clustered Server**

  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.



- **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.



Note: To view the topology, see "Topology" on page 332.

Chapter 23

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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

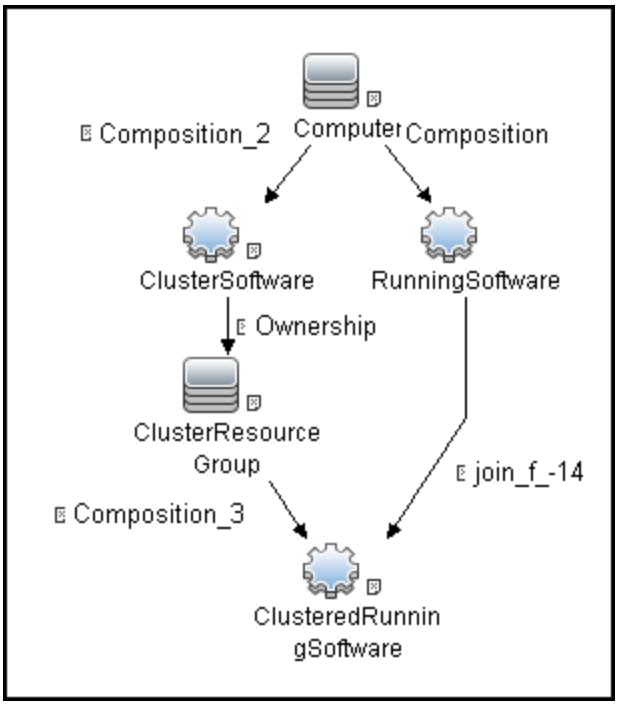
Note: To widen the scope of the discovery, the user should update the Trigger TQL Query and the Input TQL Query by adding the appropriate CIT names to the parameters. No additional changes are required.

Merge Clustered Software Job

This section includes:

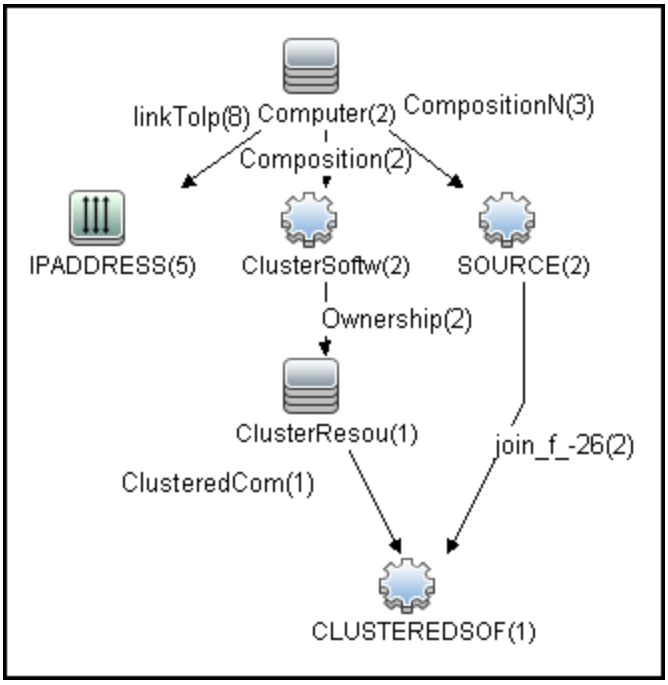
Trigger TQL Query

The following graphic shows the Trigger TQL Query for merging clustered software.


Composition_2
Computer
Composition
ClusterSoftware
RunningSoftware
Ownership
ClusterResource Group
join_f_-14
Composition_3
ClusteredRunning 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 CIs

Composition

Node

RunningSoftware

Used Scripts

mergeClusteredSoftware.py

Created/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 24

Microsoft Cluster Discovery

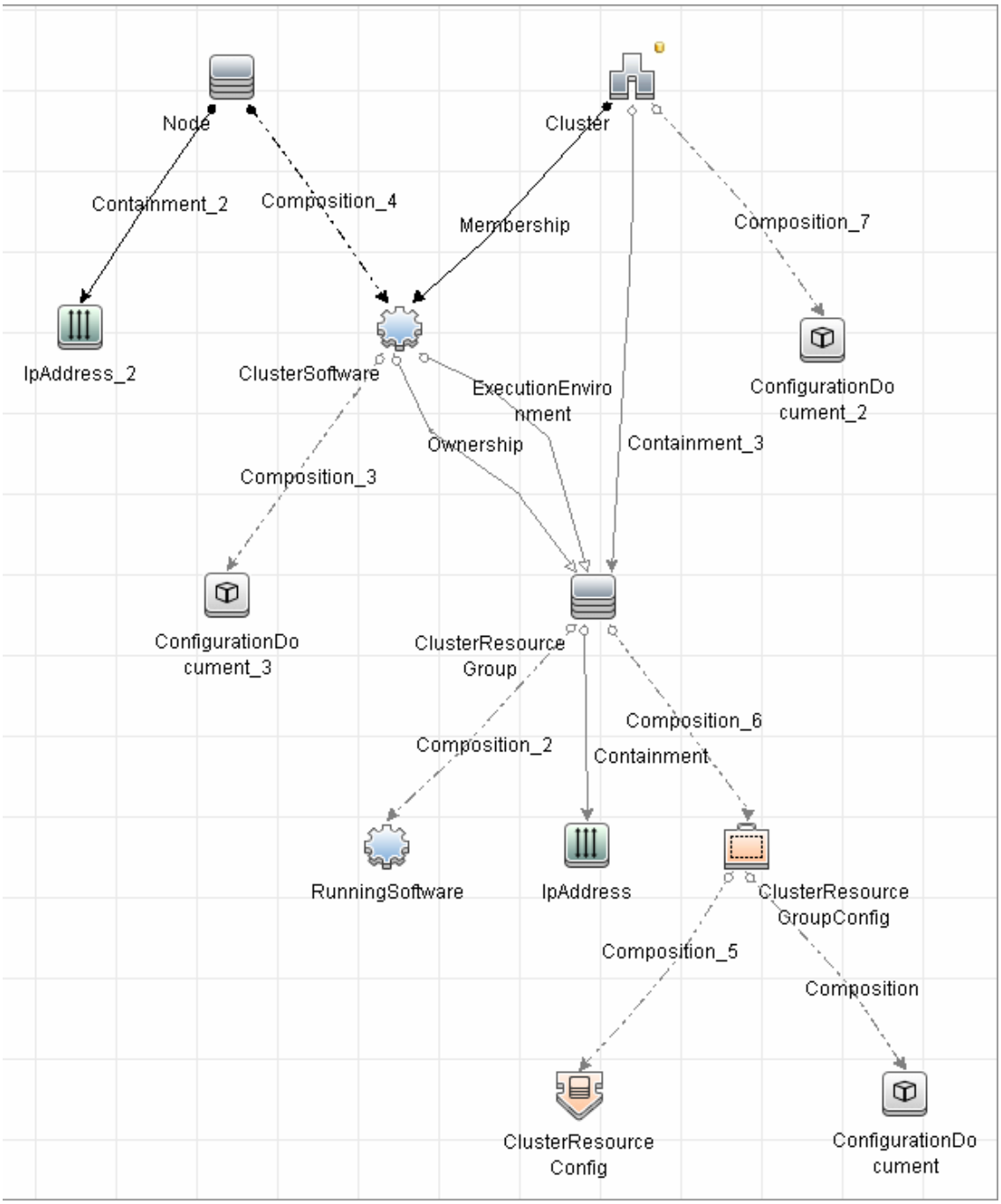
This chapter includes:

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 347.



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" on page 82.

2. **Run the discovery**

   Activate the jobs in the jobs in the **Microsoft Cluster** module in the following order:

   - **Network – Basic** (Host Connection by Shell or Host Connection by PowerShell)
   - **Network – Host Resources and Applications**
   - **Cluster – Microsoft Cluster** (MS Cluster by NTCMD or UDA)

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

MS Cluster by NTCMD or UDA Job

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 345.

Chapter 25

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 next page.

See also:

- "Components of the Network Load Balancing Architecture" on page 361
- "Glossary" on page 362

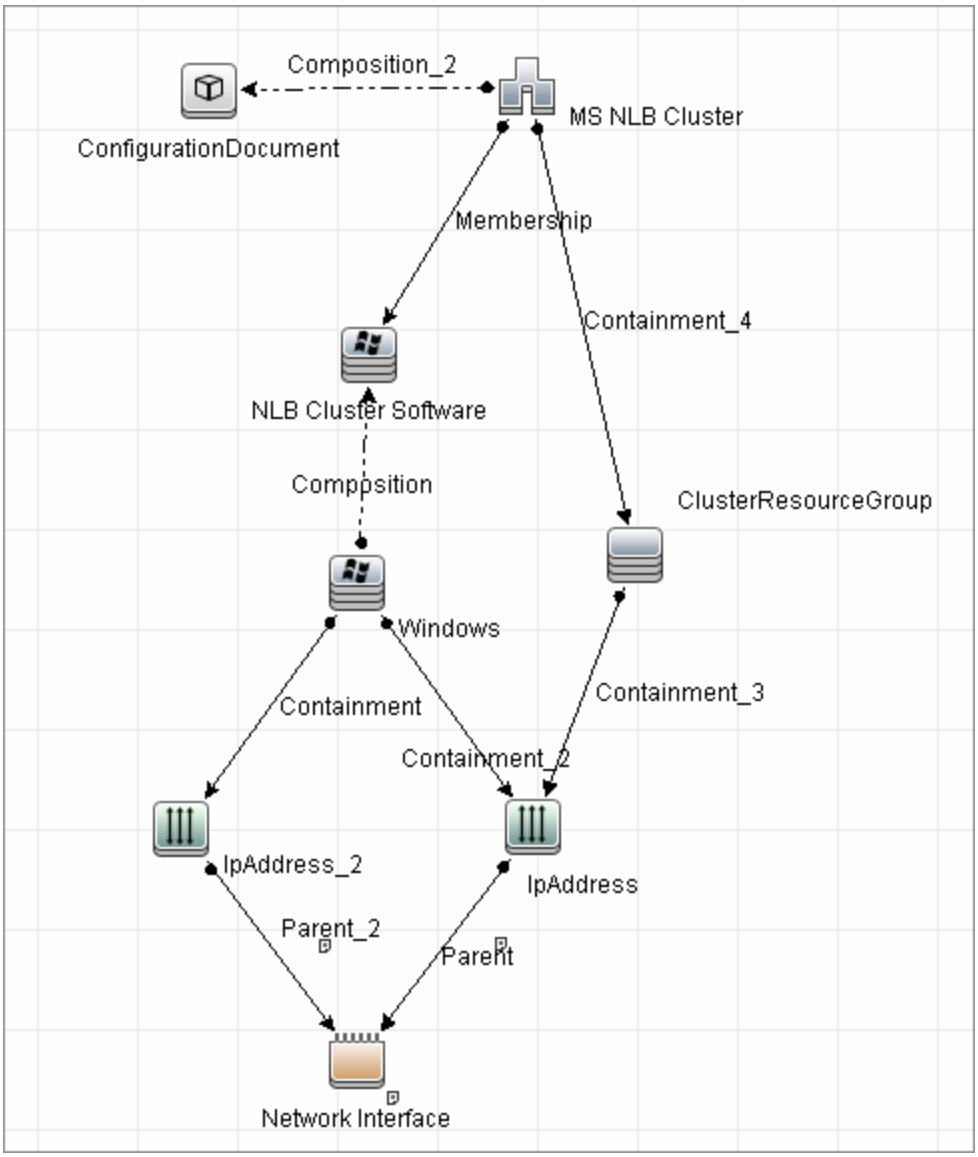
Supported Versions

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 "Discovered CITs" on page 356.



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.

   For credential information, see "Supported Protocols" on page 82.

   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 "Discovery Control Panel" 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** job to connect to the host by NTCMD and retrieve the MS NLB Cluster topology. For job details, see "MS NLB by NTCMD Job" on page 354.

   For details on the discovery mechanism, see "Discovery Mechanism" on page 354.

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 Job

This section includes:

Discovery Mechanism

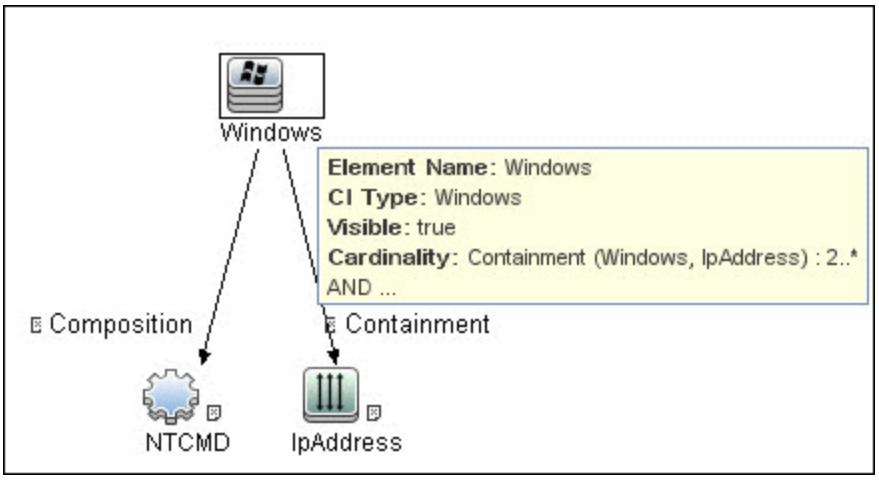
DFM triggers on Windows machines with more than one (two or more) IP addresses, and collects information using the **nlb.exe** command line utility. (In earlier versions of the Windows 2000 family, **wlbs.exe** is used.) These utilities enable the retrieval of all NLB-related information. For details, see "MS NLB by NTCMD Adapter" on page 357.

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 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** 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" on next page.

Views

- Microsoft NLB topology

Discovered CITs

- **Composition**
- **ConfigurationDocument.** For details, see "MS NLB by NTCMD Adapter" on next page.
- **Containment**
- **IpAddress**
- **Membership**
- **MS NLB Cluster.** For details, see "MS NLB by NTCMD Adapter" on next page.
- **NLB Cluster Software.** For details, see "MS NLB by NTCMD Adapter" on next page.
- **Node**

Note: To view the topology, see "Topology" on page 351.

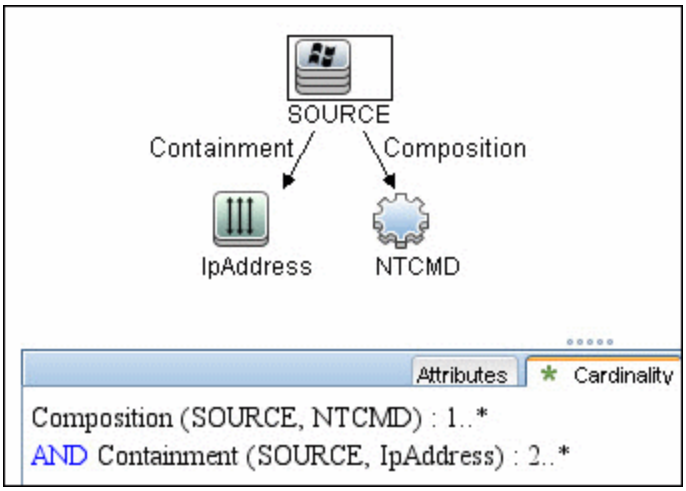
MS NLB by NTCMD Adapter

This section includes:

Input Query

- **Input Query**

  NTCMD running on a Windows machine with at least two IP addresses:



- **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

Links

Start Node	Start Node Cardinality	Link 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

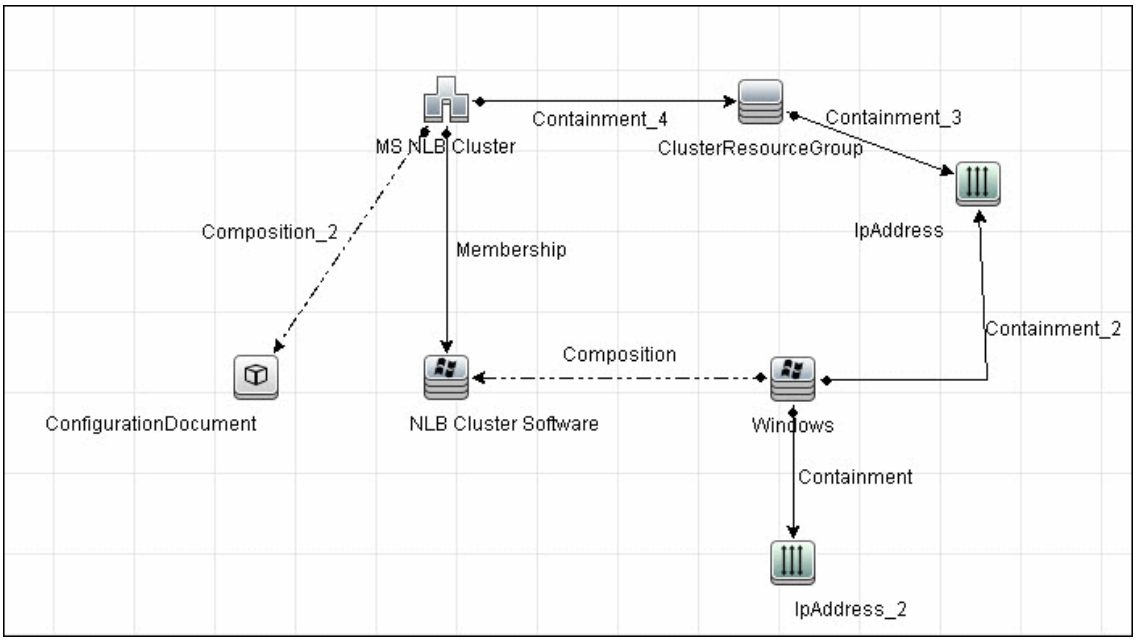
Links

Start Node	Start Node Cardinality	Link 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
```

Links

Start Node	Start Node Cardinality	Link 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.
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.

Chapter 26

Sun 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

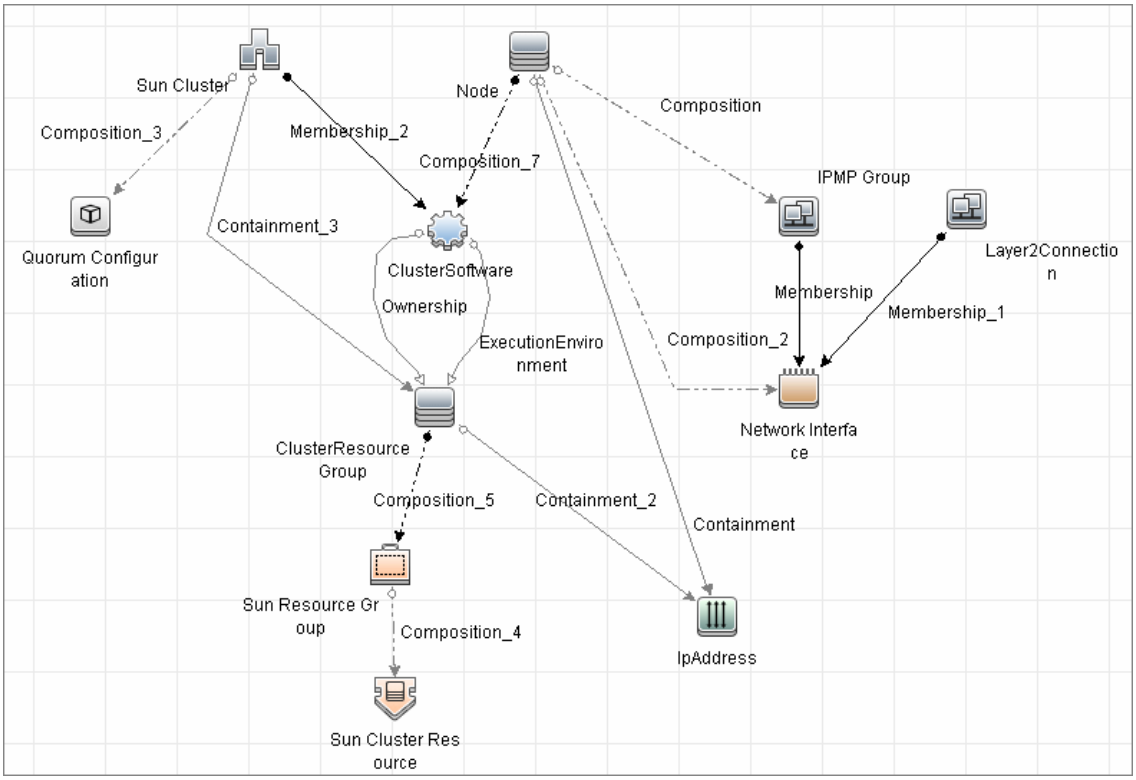
The **Sun Cluster** package supports Sun Cluster 3.2. Support for older versions of Sun Cluster has not been verified.

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 Sun Cluster discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 368.



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" on page 82.
   - Set up permissions for users performing Sun Cluster discovery to run clustering commands (**scrgadm**, **scstat**, **scconf**, and so on). For a full list of commands see "Sun Cluster Discovery Commands" on page 369.
2. **Run the discovery**

   For details on running jobs, refer to "Discovery Control Panel" 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 Resources and Applications by Shell** job to discover processes on the target machines.

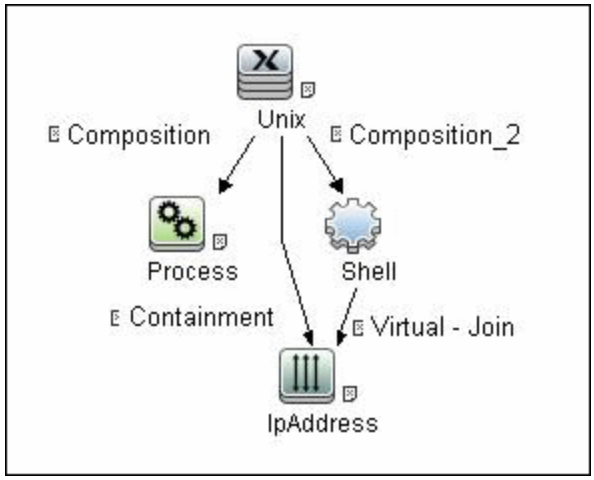
   d. Run the **Sun Cluster by Shell** job to discover the Sun Cluster topology. For job details, see "Sun Cluster by Shell Job" below.

Sun Cluster by Shell Job

This section includes:

Trigger Query

- **Trigger query**:



- **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`

Adapter

- **Input Query**

  This query contains only one Shell CI:

- **Created/Changed Entities**

Added CI Types	▪ Sun Cluster ▪ Sun Resource Group ▪ Sun Cluster Resource ▪ IPMP Group
Added valid links	▪ Node - composition > IPMP Group ▪ IPMP Group - membership > Network Interface
Added views	Sun Cluster Topology view
Added scripts	▪ sun_cluster_by_shell.py ▪ solaris_networking.py
Added adapters	Sun_Cluster_by_Shell
Added jobs	Sun Cluster by Shell
Added trigger query	shell_on_solaris_cs
Added module	Sun Cluster.xml

Used Scripts

- **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 364.

Sun Cluster Discovery Commands

This section includes the Sun clustering commands:

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 (oracle1:oracle1-zfs{kvsdb2}) Res enabled: True

Command	/usr/cluster/bin/scrgadm -pvv
	(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.

Command	/usr/cluster/bin/scrgadm -pvv
	(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 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

Command	/usr/cluster/bin/scrgadm -pvv
	(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 (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.

Command	/usr/cluster/bin/scrgadm -pvv
	(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 (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

Command	/usr/cluster/bin/scrgadm -pvv
	(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 (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

Command	/usr/cluster/bin/scrgadm -pvv
	(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

Command	/usr/cluster/bin/scrgadm -pvv
	(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. (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

Command	/usr/cluster/bin/scrgadm -pvv
	(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. (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

Command	/usr/cluster/bin/scrgadm -pvv
	(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: 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

Command	/usr/cluster/bin/scrgadm -pvv
	(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 ...
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

Command	/usr/cluster/bin/scrgadm -pvv
	▪ 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 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 27

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.

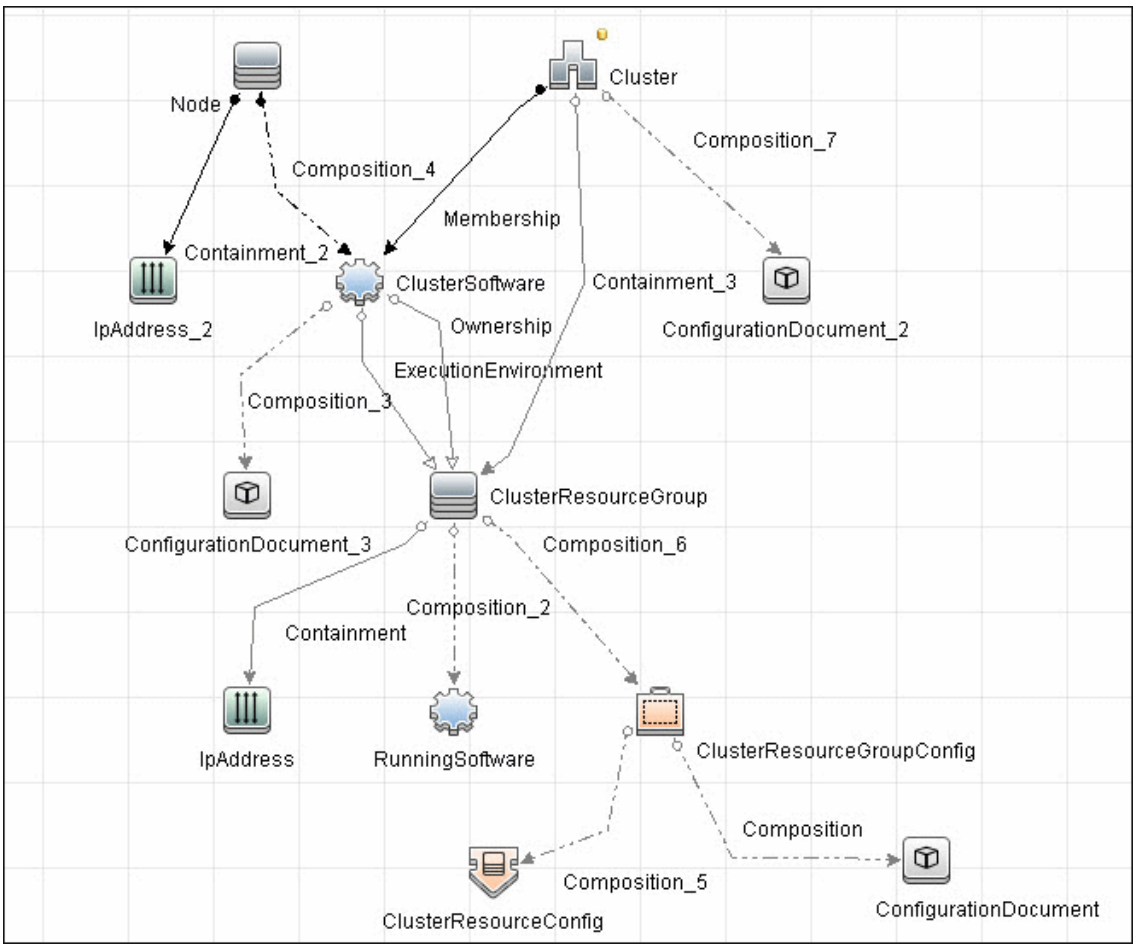
Supported Versions

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 "Discovered CITs" on page 398.



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" on page 82.

2. **Run the discovery**

   For details on running jobs, refer to "Discovery Control Panel" 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 Resources and Applications by Shell** job.

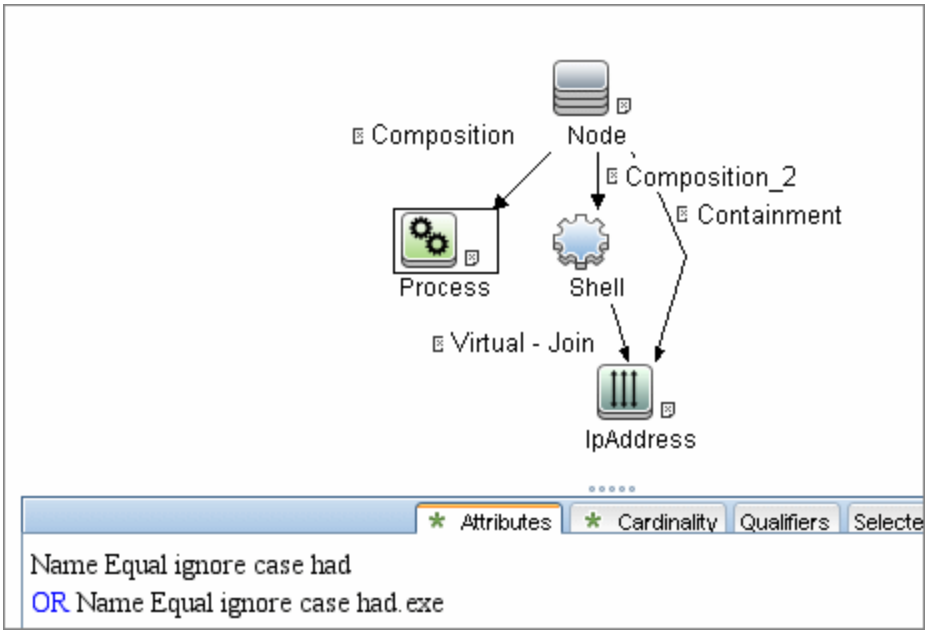
   c. Run the **Veritas Cluster by Shell** job. For job details, see "Veritas Cluster by Shell Job" below.

Veritas Cluster by Shell Job

This section includes:

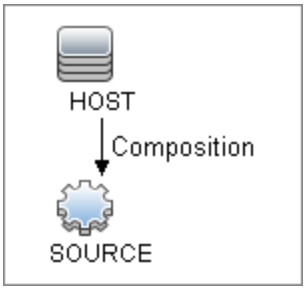
Trigger Query

- **Trigger query**:



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 395.

Part IV: Databases

Chapter 28

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 403
- "DB Connection by WMI Job" on page 407

Supported Versions

This discovery solution supports the following database servers:

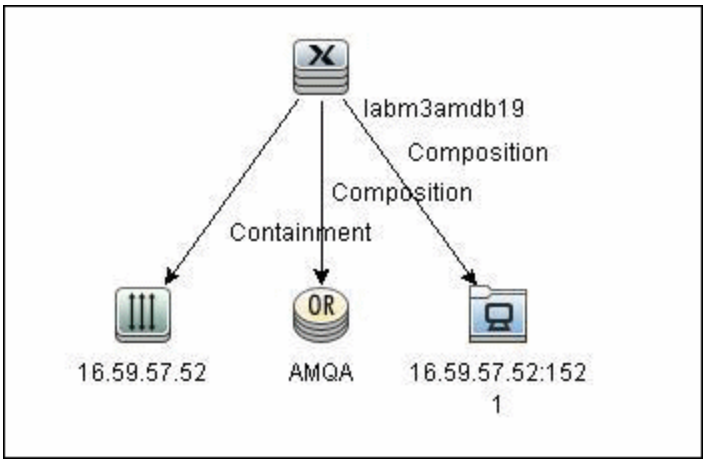
- Oracle 9i, 10g, 11g
- Microsoft SQL Server 2000, 2005, 2008
- IBM DB2 8.x and 9.x

Topology

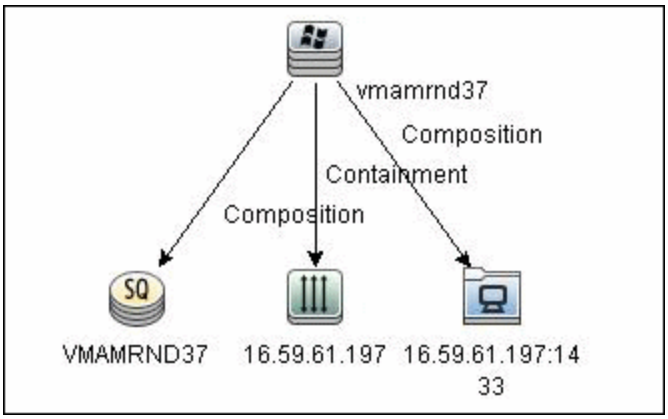
The following images displays the topology of the Database Connections by Host Credentials discovery with sample output:

Note: For a list of discovered CITs, see "DB Connection by Shell Job" on page 403 and "DB Connection by WMI Job" on page 407.

Oracle


labm3amdb19
Composition
Composition
Containment
16.59.57.52
AMQA
16.59.57.52:1521

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" on page 82.

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 "Discovery Mechanism" on next page.
   e. Run the **DB Connections by WMI** job. For details, see "DB Connection by WMI Job" on page 407.

DB Connection by Shell Job

This section includes:

Discovery Mechanism

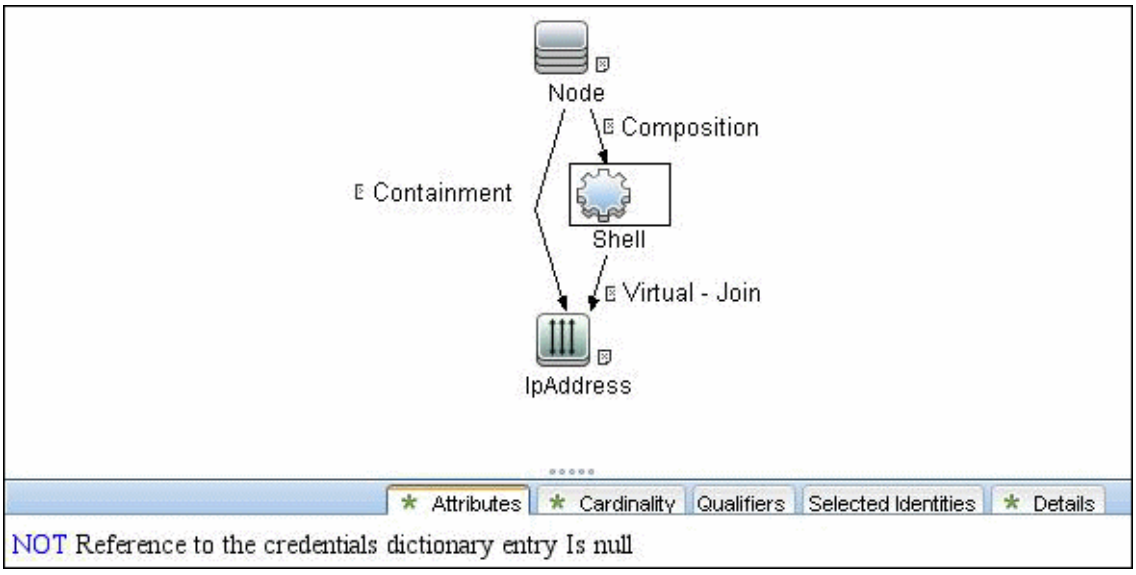
This discovery job attempts to identify configured databases on a host using a Shell client (NTCMD/SSH/Telnet). Once connected, the job creates a list of running processes and server ports associated with each process. On Microsoft Windows operating systems, this job adds a list of installed Windows 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 "Domain Credential References" in the *HP Universal CMDB Data Flow Management 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

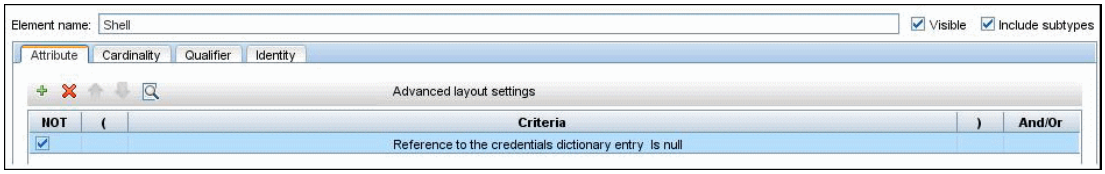


Adapter

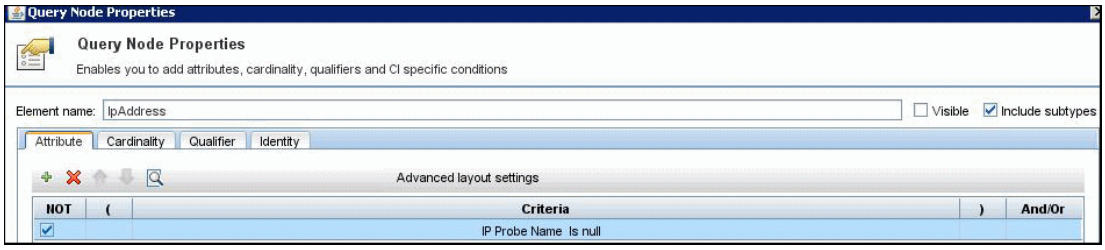
This job uses the **Database Connections by Shell** adapter

- **Input query:** None
- **CI Attributes conditions:**

- **Shell attributes**:



- **IpAddress attributes**:



- **Adapter Parameters**

Parameter	Description
discover_db2. true	DFM discovers IBM DB2 database servers.
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**
- **DB2**
- **IpAddress**
- **IpServiceEndpoint**

- **Node**
- **Oracle**
- **SQL Server**
- **Unix**
- **Windows**

Note: To view the topology, see "Topology" on page 401.

DB Connection by WMI Job

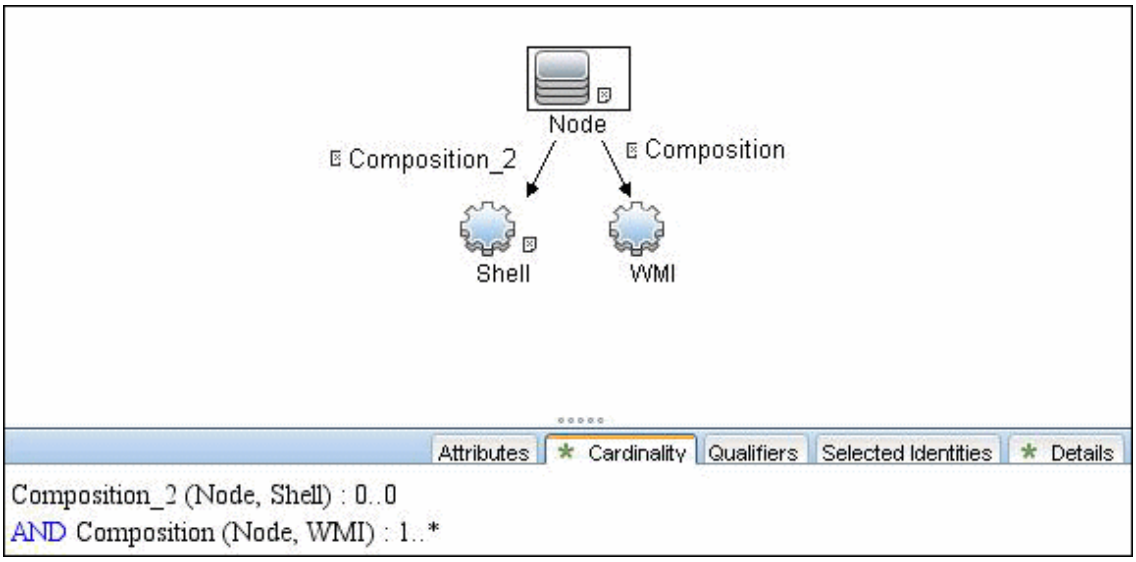
This section includes:

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



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 401.

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 29

IBM DB2 Database Discovery

This chapter includes:

Supported Versions

This discovery supports the following versions:

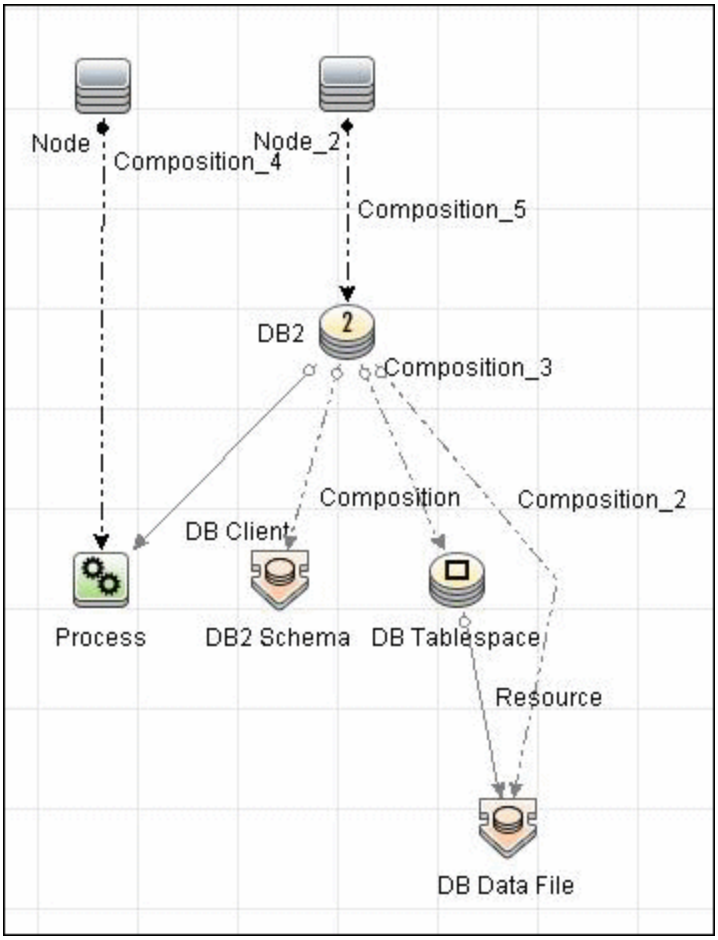
IBM DB2 Universal Database (UDB) versions 8.2, 9.1, 9.5, 9.7

Topology

The following image depicts the topology of the IBM DB2 Server view.

This view shows a host on which an IBM DB2 Server and DB2 Schema are installed, the processes that communicate with the server (connected by DB Client links), and the DB tablespaces.

Note: For a list of discovered CITs, see "DB2 Universal Database Connection by SQL Job" on page 412, "DB2 Topology by SQL Job" on page 412, and "Databases TCP Ports Job" on next page.



How to Discover IBM DB2 Databases

This module discovers IBM DB2 Server databases and their components on the network, and includes the following steps.

This task includes the following steps:

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

   IBM DB2 Server uses the Generic DB Protocol (SQL).

   In the Database Type box, choose **db2**.

   For credential information, see "Supported Protocols" on page 82.

2. **Prerequisites - Miscellaneous**

   - Verify the user name, password, and port used by IBM DB2 Server.
   - To perform an IBM DB2 discovery, copy the following files from the installation folder on the IBM DB2 machine to the Data Flow Probe machine:
     - **db2java.zip**
     - **db2jcc.jar**
     - **db2jcc_license_cisuz.jar**
     - **db2jcc_license.jar**

     Place the files in the following folder: **<hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\db\db2**. Restart the Data Flow Probe.

3. **Run the discovery**

   Activate the jobs in the following order:

   - **Databases TCP Ports**
   - **DB2 Universal Database Connection by SQL**
   - **DB2 Topology by SQL**

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

Databases TCP Ports Job

Discovered CITs

You can view discovered CITs for an adapter in the Adapter Manager module. 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 Statistics table in the **Details** tab. For details, see "Statistics Results Pane" in the *HP Universal CMDB Data Flow Management Guide*.

- **Composition**
- **Containment**
- **IpAddress**
- **IpServiceEndpoint**
- **Node**

Note: To view the topology, see "Topology" on page 410.

DB2 Universal Database Connection by SQL Job

Discovered CITs

You can view discovered CITs for an adapter in the Adapter Manager module. For details, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

- **DB2**
- **Composition**

Note: To view the topology, see "Topology" on page 410.

DB2 Topology by SQL Job

Discovered CITs

You can view discovered CITs for an adapter in the Adapter Manager module. 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 Statistics table in the **Details** tab. For details, see "Statistics Results Pane" in the *HP Universal CMDB Data Flow Management Guide*.

- **DB Data File**
- **DB Tablespace**
- **DB2**
- **DB2 Schema**
- **IpAddress**
- **Node**
- **Process**
- **Composition**
- **Containment**
- **DB Client**
- **Resource**

Note: To view the topology, see "Topology" on page 410.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for IBM DB2 discovery.

- DB2 databases are not discovered by DB connections by a WMI job because DB2 information is not available in the Windows registry.

Chapter 30

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

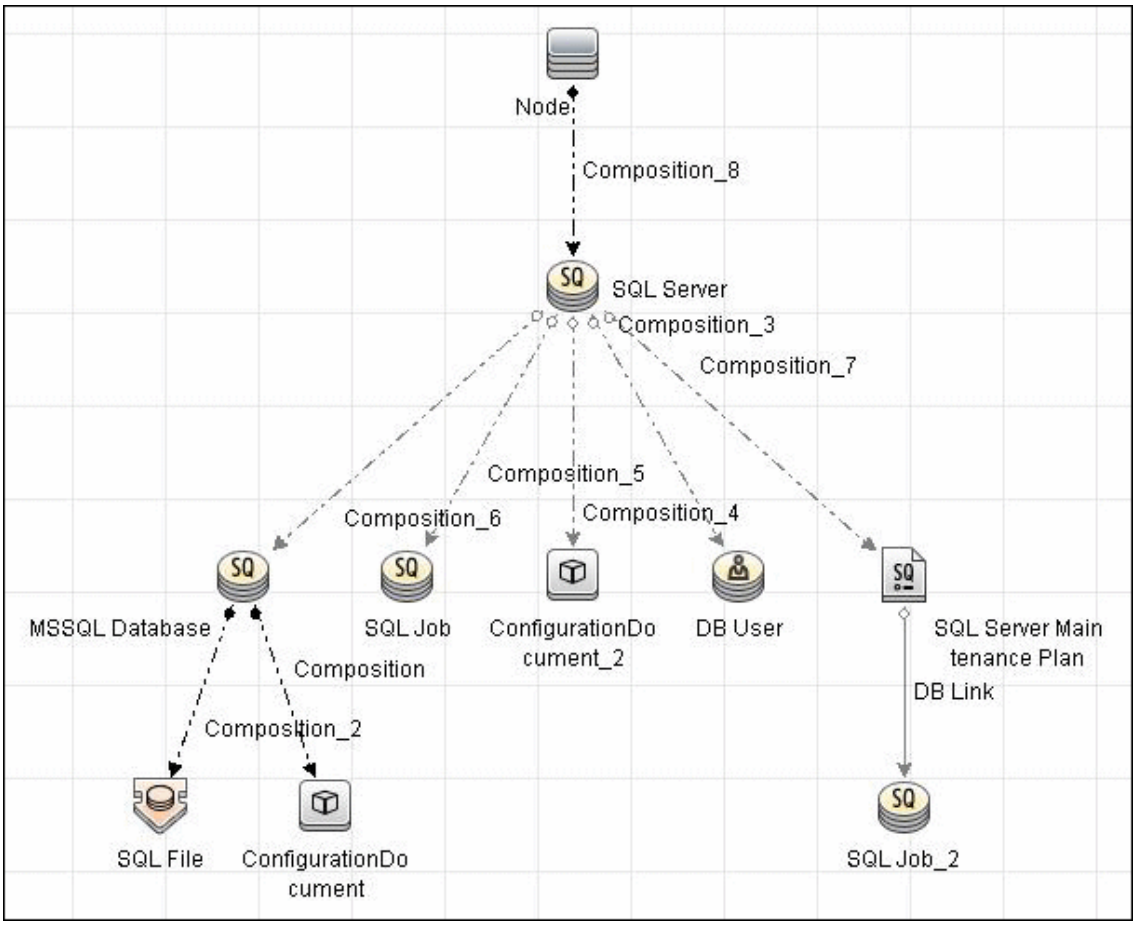
This discovery supports Microsoft SQL Server versions 2000, 2005, and 2008.

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 418.



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" on page 82.

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 Discovery Control Panel 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 "Discovery Control Panel" 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 Resources and Applications by Shell**
   - **Host Resources and Applications by WMI**
   - **DB connection by Shell**
   - **DB connection by WMI**

   For details on running jobs, refer to "Discovery Control Panel" 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.

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 Statistics table in the Details tab. For details, see "Statistics Results Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Note: To view the topology, see "Topology" on page 416.

SQL Server by OS Credentials Discovery

DFM can discover MS SQL Server CIs using operating system (OS) credentials. DFM 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 DFM found this process, generic running software with a **MSSQL DB** value in the **name** attribute was reported to UCMDB.

Data Flow Probe can report multiple SQL Server instances, each of them linked by a dependency link to its own **sqlservr.exe** process.

DFM 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. DFM 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 DFM reports the generic running software CI instead.

- **Using Windows Services**. DFM checks existing services for those that include **sqlservr.exe** in the command line and extracts the instance name from the service name (since the service name reflects the instance name).

Chapter 31

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

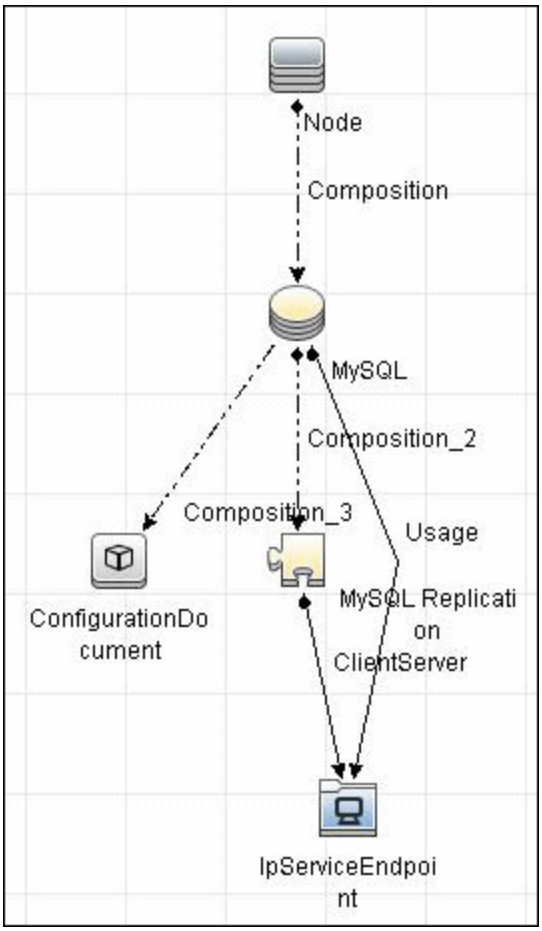
This discovery supports the following:

- MySQL versions 4.x, 5.x, 6.0
- Operating systems: Windows, Solaris, and Linux

Topology

Note: For a list of discovered CITs, see "Discovered CITs" on page 428.

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" on page 82.

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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

MySQL by Shell Job

This section includes:

Discovery Mechanism

This section explains how DFM discovers the MySQL server:

- The MySQL by Shell job connects to the remote host using Shell credentials.
- The job checks for the existence of the path of the MySQL configuration file by executing the following command:

  ```
  mysqld --verbose --help
  ```

- 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**

- 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:

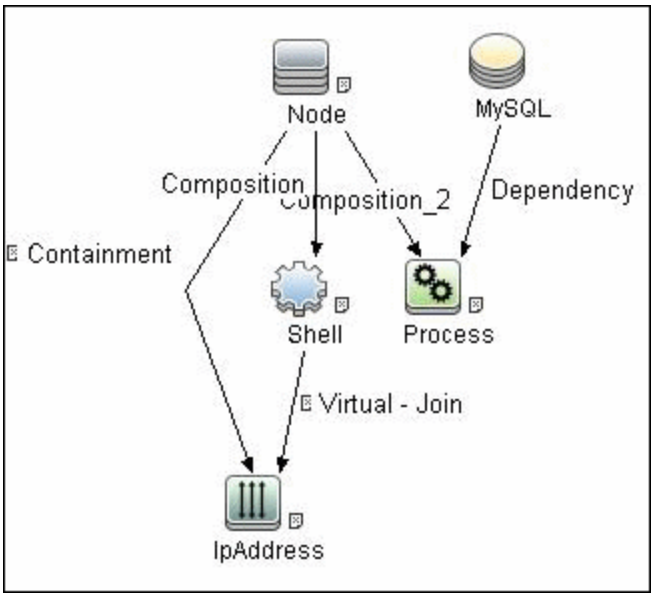
  ```
  mysqld-nt.exe --defaults-file=C:\hp\UCMDB\DataFlowProbe\MySQL\my.ini
  DDM_Probe_DB
  ```

- 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 "CIT Attributes" on page 426.

- The job creates the MySQL CIs with appropriate attribute values and relationships.
- 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.
- 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

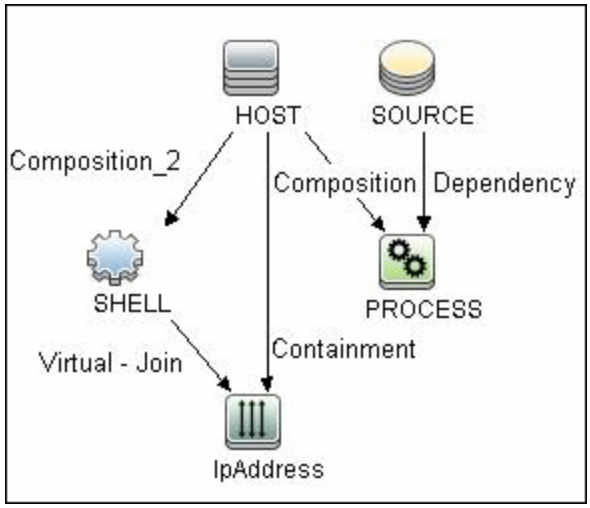
- **MySQL**
  - **server_id**. The server ID is used in the replication job and must be unique for each server.
  - **database_datadir**. Path to the database root (**datadir** in the configuration file).
  - **database_max_connections**. The maximum number of concurrent sessions allowed by the MySQL 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**



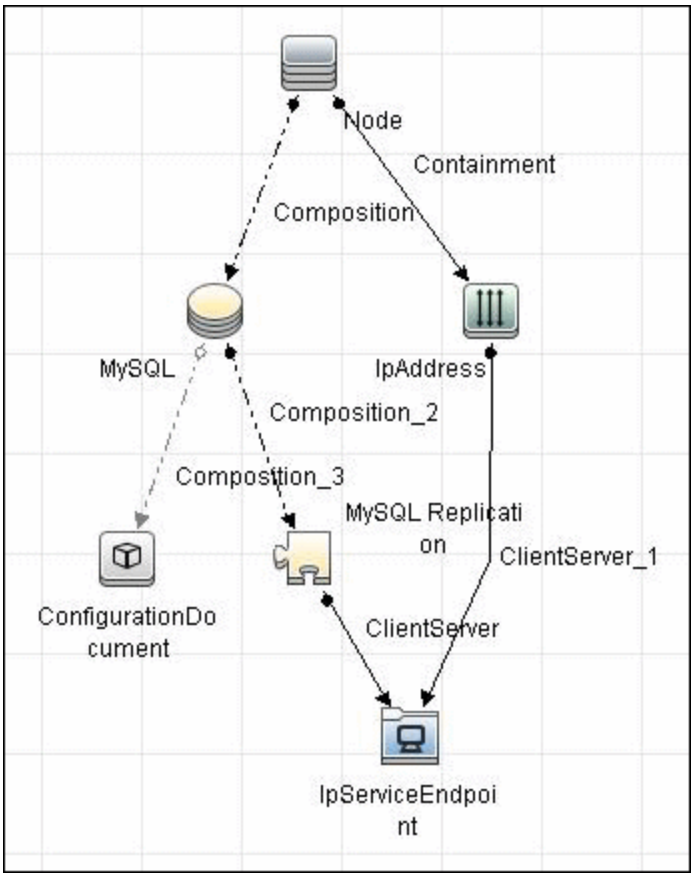
- **Triggered CI Data**

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.

For details, see "Discovered CITs Pane" in the *HP Universal CMDB Data Flow Management Guide*.



- **ClientServer**
- **Composition**
- **ConfigurationDocument**
- **Containment**
- **IpAddress**
- **IpServiceEndpoint**
- **MySQL**
- **MySQL Replication**
- **Node**

Note: To view the topology, see "Topology" on page 422.

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 32

Oracle Database Server Discovery

This chapter includes:

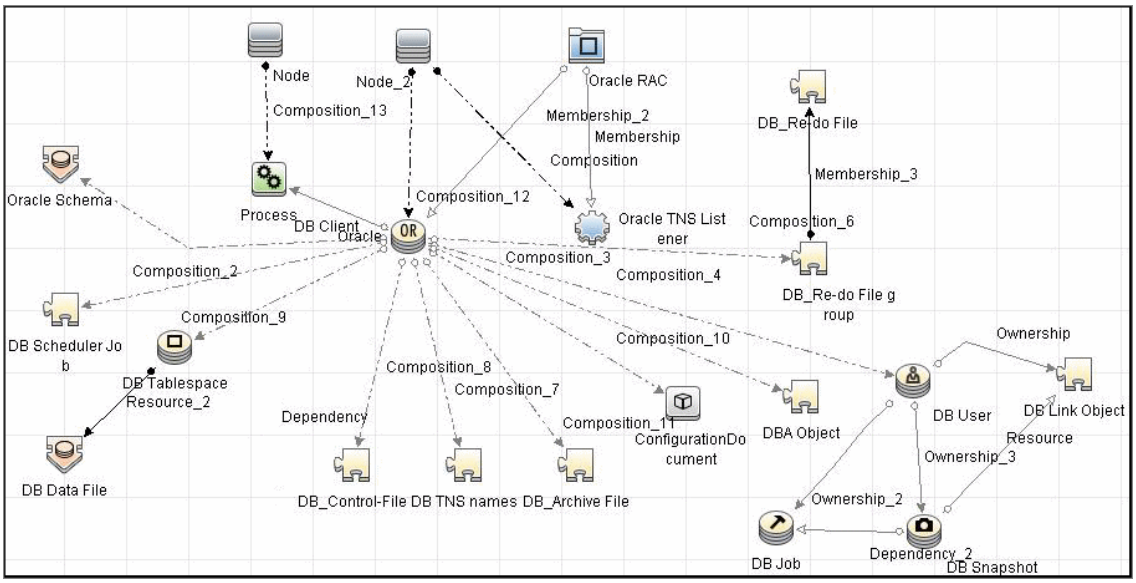
Supported Versions

This discovery supports Oracle 8, 9, 10, 11g.

Topology

The following image displays the topology of the Oracle Database Server discovery:

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



How to Discover Oracle Databases

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 discovery uses the Generic DB Protocol (SQL).

   For credential information, see "Supported Protocols" on page 82.

2. **Prerequisite - Verify user on Oracle database server**

   Run **Databases TCP Ports**. Verify the user name, password, and port used by the Oracle Database Server.

3. **Run the discovery**

   Activate the jobs in the following order:

- **Databases TCP Ports**
- **Oracle Database Connection by SQL**
- **Oracle Topology by SQL**

> **Note:** Due to the large amount of data reported by the 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, refer to "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Oracle Database Server Discovery

Created/Changed Entities

The following attributes are updated:

- **version**
- **database_dbtype**
- **database_dbsid**
- **application_port**

Discovered CITs

- **ownership**
- **dbjob**
- **dbuser**
- **process**
- **dbclient**
- **dblinkobj**
- **dbsnapshot**
- **dbdatafile**
- **dbtablespace**
- **db_controlfile**
- **db_redofile**
- **db_redofilegroup**
- **db_archivefile**
- **oracle**
- **dbschedulerjob**

- **rac**

Note: To view the topology, see "Topology" on page 431.

Chapter 33

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

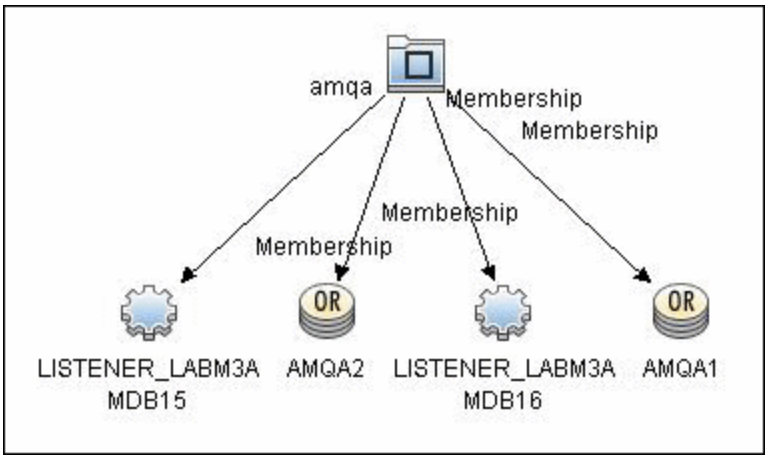
This discovery supports Oracle DB 10 and 11.

Topology

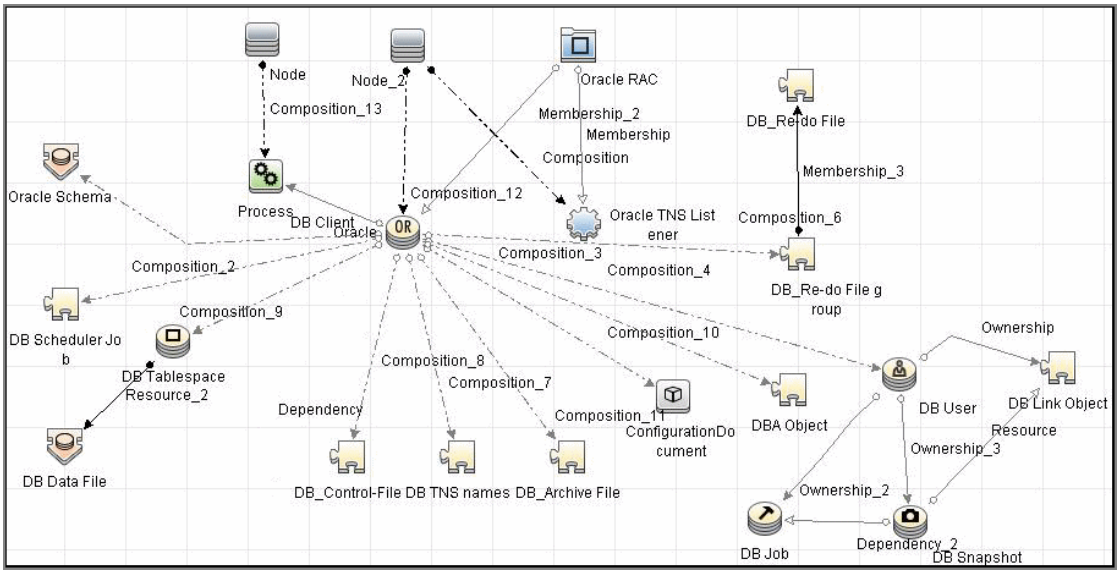
The following images display sample output of the Oracle RAC discovery topology.

Note: For a list of discovered CITs, see "Oracle Listeners by Shell Job" on page 437 and "Oracle RAC Topology by Shell Job" on page 440.

- **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" on page 82.

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 Resources and 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 440. For details on undiscovered elements, see "Troubleshooting and Limitations" on page 444.

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

Oracle Listeners by Shell Job

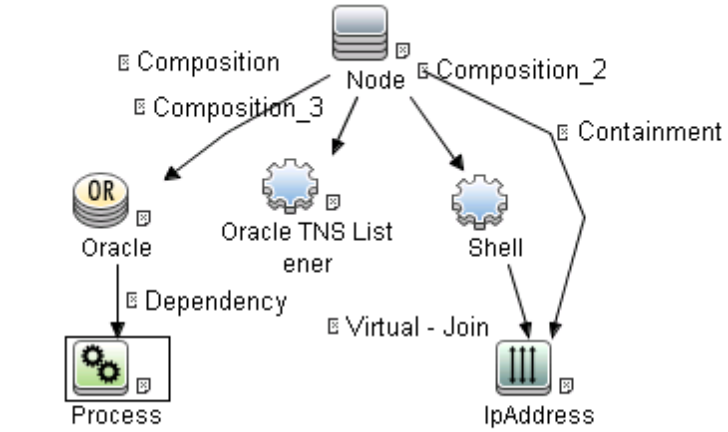
This section includes:

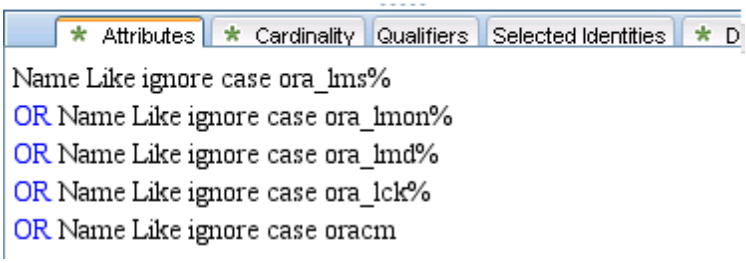
Discovery Mechanism

This job triggers on Oracle databases that have RAC related processes. The job:

- Connects to the remote host by Shell.
- Checks for the **ORACLE_HOME** environment variable.
- If the variable is not defined, the job takes the **ORACLE_HOME** value from the job adapter (if defined).
- Reads the **Oracle TNS listener** configuration file, stored in **$ORACLE_HOME/network/admin/listener.ora**, and performs further parsing.
- Retrieves a full list of IP addresses to which this particular listener is listening.
- Checks for listener status using the **$ORACLE_HOME/bin/lsnrctl** status.
- Retrieves known services and listener status from the output.

Trigger Query

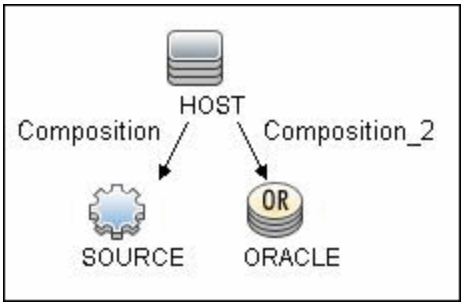




Adapter

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 435.

Oracle RAC Topology by Shell Job

This section includes:

Discovery Mechanism

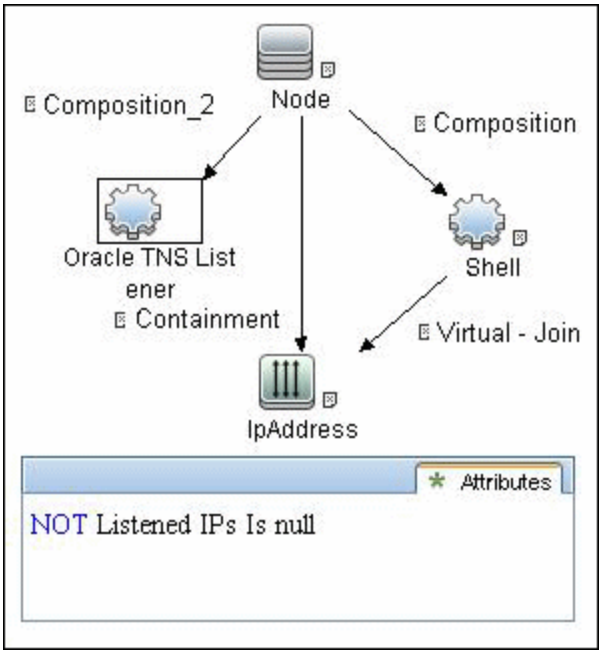
This job:

- Connects to the remote host by Shell.
- Checks for the **ORACLE_HOME** environment variable.
- If it is not defined, the job uses the **OracleHome** value from the job adapter.
- Retrieves RAC parameters such as Service Name and Nodes from the **$ORACLE_HOME/network/admin/tnsnames.ora** file.
- 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.

- Retrieves the full list of Listened IPs from the input query, for all listeners matching the query.
- 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 (:).
- 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.
- Creates the RAC CI.

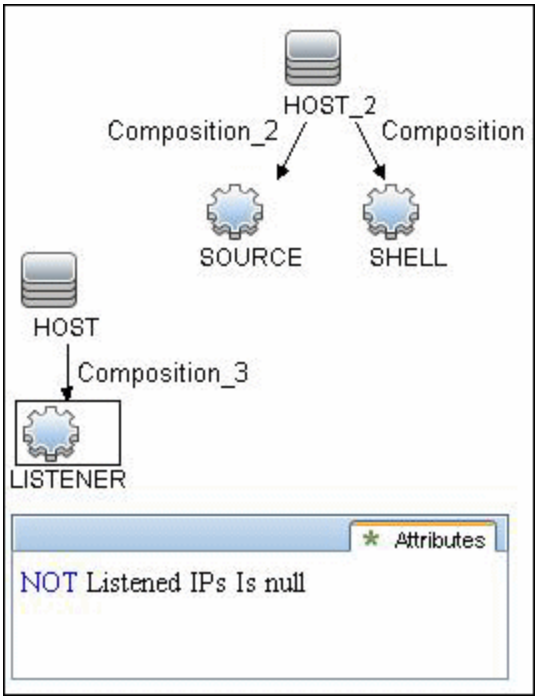
Trigger Query



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 435.

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 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.

Chapter 34

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.

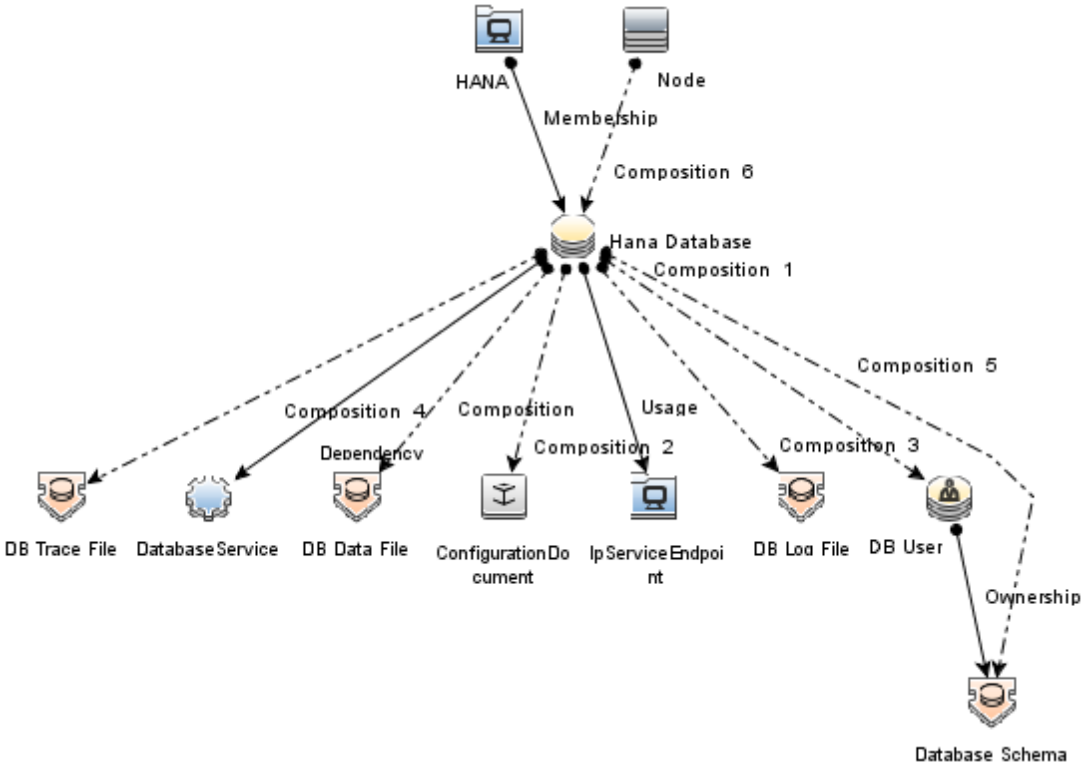
Supported Versions

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 450.



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 - Connectivity and user store**
   a. Shell connectivity to a HANA Database Node.
   b. Properly configured user store containing one user key for each HANA Database instance being discovered.

      The user key name should follow the pattern **cmdb<SID>**. For example, for sid HHP: **cmdbHHP**.

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" on page 82.

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 Resources and Applications by Shell** job to discover the resources of the target host, including HANA Database software and relevant processes.
   d. Run the **HanaDb by Shell** job to discover the topology of the target HANA Database.

HanaDb by Shell Job

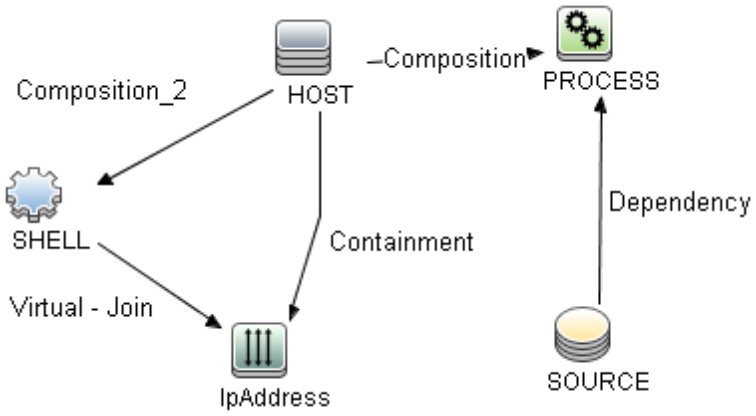
This section includes:

Adapter

This job uses the **HanaDb_by_Shell** adapter.

Trigger Query

Name: hanadb



Node Name	Condition
HanaDb	None
IpAddress	NOT IP Probe Name Is null
Process	Name Like ignore case %hdb.sap%
Shell	None
Node	None

Parameters

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

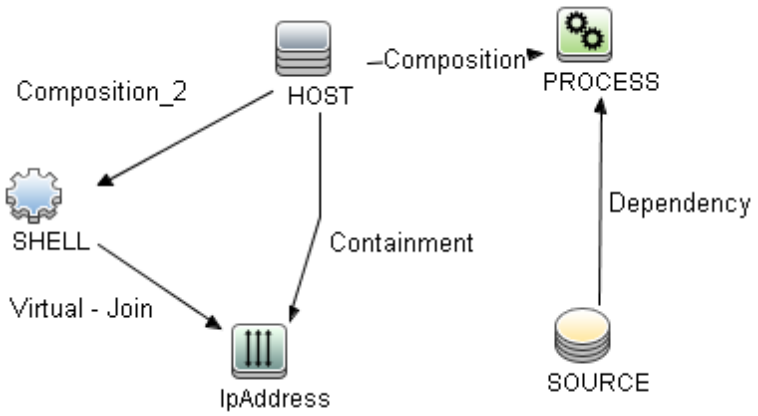
HanaDb_by_Shell Adapter

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	None

Triggered CI Data

Name	Value
Protocol	${SHELL.root_class}
credentialsId	${SHELL.credentials_id}
dbport	${SOURCE.application_port:}
dbsid	${SOURCE.name}
processParams	${PROCESS.process_parameters:}
processPath	${PROCESS.process_path:}
ip_address	${SHELL.application_ip}

Used Scripts

- **command.py**
- **db.py**
- **db_builder.py**
- **db_platform.py**
- **entity.py**
- **fptools.py**
- **hana.py**
- **hana_discoverer.py**
- **hanadb_by_shell.py**
- **iteratortools.py**

Discovered CITs

- **Composition**
- **ConfigurationDocument**
- **Containment**
- **Database Schema**
- **DB Data File**
- **DB User**
- **DbLogFile**
- **DbTraceFile**

- **Dependency**
- **HanaDatabase**
- **IpAddress**
- **IpServiceEndpoint**
- **Node**
- **Ownership**
- **RunningSoftware**
- **Usage**

Global Configuration Files

Name: _HanaDb by Shell.xml.

> **Note:** This is a configuration file for internal usage only, and should not be changed.

Discovery Flow

1. **Calculate Paths**

   The path of the HDB Daemon process discovered by Application Signature is analyzed, and the following entities are calculated:

   - Hana Database sid
   - hdbsql path

2. **Discover HANA Database Topology**

   The command **hdbsql** is used to read relevant information from the database, such as:

   - Instance number
   - Version, start time, and database name
   - Config files
   - Config file contents
   - Schemas
   - Users
   - Data files
   - Log files
   - Trace files
   - Trace files path

Output Samples

Obtain Instance Number Information of Installed HANA Database

Command

```
/usr/sap/hdbclient -j -U cmdbHPS "select value from m_host_information
where key = 'sapsystem'"
```

Output

```
VALUE
"43"
1 row selected (0 usec)
```

Get Version, Start Time, and Database Name

Command

```
/usr/sap/hdbclient -j -U cmdbHPS "select database_name, host, start_
time, SECONDS_BETWEEN('1970-01-01', START_TIME) as start_time_in_
seconds, version from m_database"
```

Output

```
DATABASE_NAME,HOST,START_TIME,START_TIME_IN_SECONDS,VERSION
"HPS","ks294","2011-12-02
16:03:10.458000000",1322841790,"1.00.16.354058"
1 row selected (0 usec)
```

Chapter 35

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.

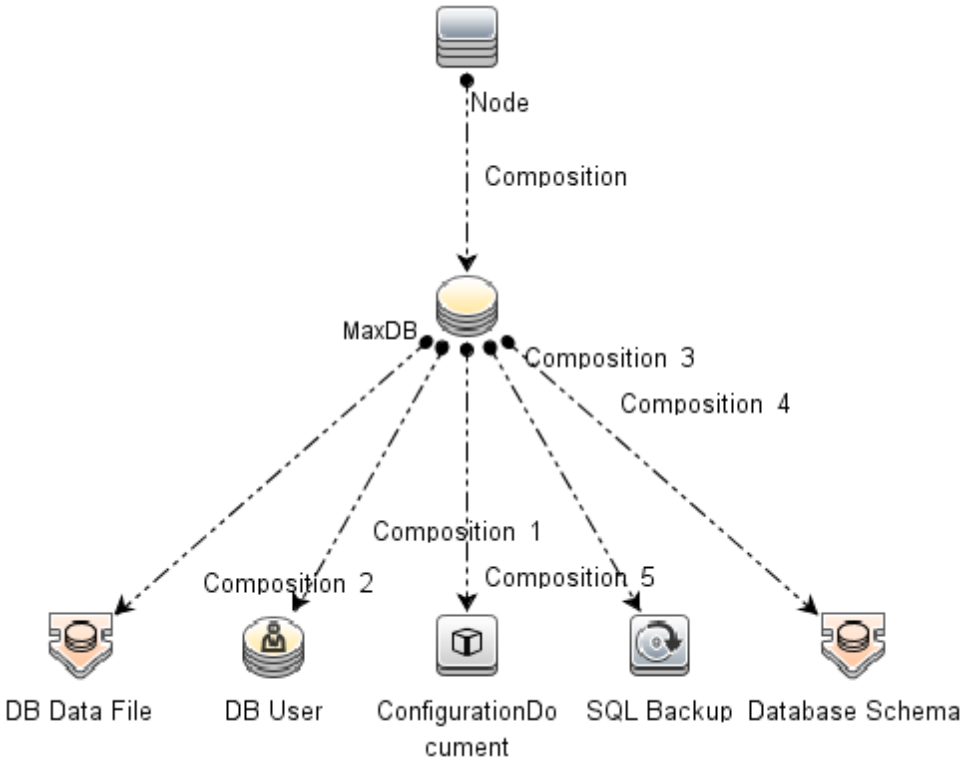
Supported Versions

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 458.



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" on page 82.

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 Resources and 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.

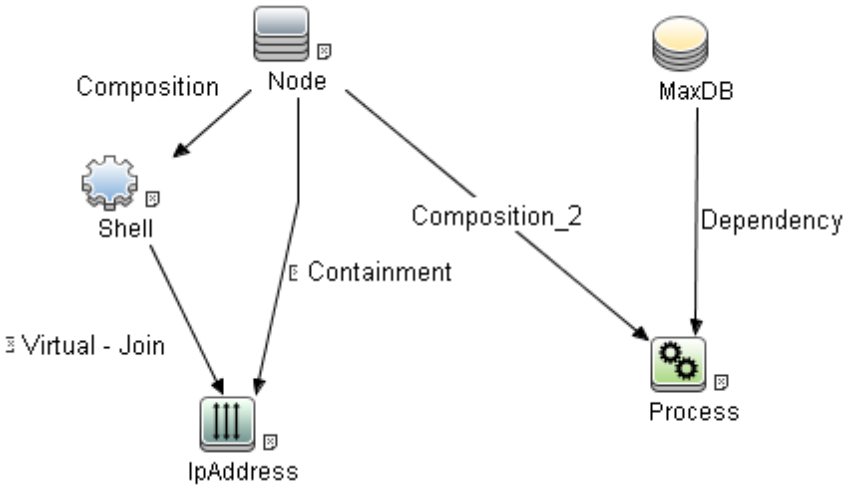
MaxDb by Shell Job

This section includes:

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.

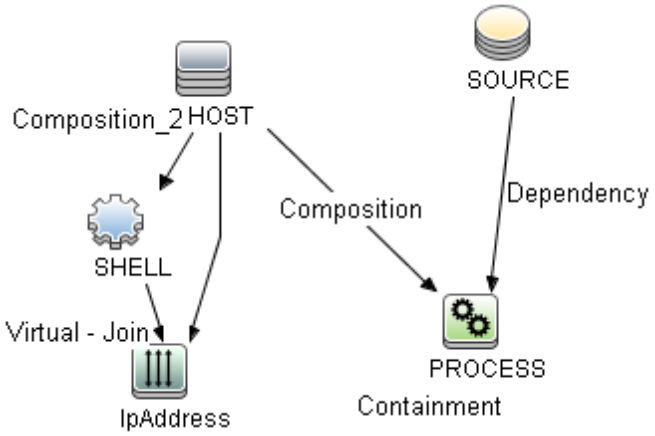
MaxDb by Shell Adapter

This section includes:

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:}
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**

Part V: Discovery Samples and Tools

Chapter 36

Discovery Tools

Overview

The Discovery Tools module contains the jobs necessary to:

- Discover document files and directories.
- Discover hosts using the **Nslookup** command on the Shell of every DNS server in the scope.
- Serve as an example of dynamically creating and using credentials for connecting to remote machines.
- Import data from external sources, for example, CSV files, properties files, and databases. For details, see "Importing Data from External Sources" on page 631.

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.

Part VI: Integrations

Chapter 37

Aperture VISTA Integration

This chapter includes:

Overview

Aperture VISTA is used to model datacenter information, including the exact location of a physical server in a rack, row, space, and floor of a datacenter. VISTA also contains detailed information about the power supply to racks and individual servers. This enables impact analysis from a power supply point of view, and with integration it becomes possible to analyze the impact of power failure on applications, business services and lines of business in UCMDB.

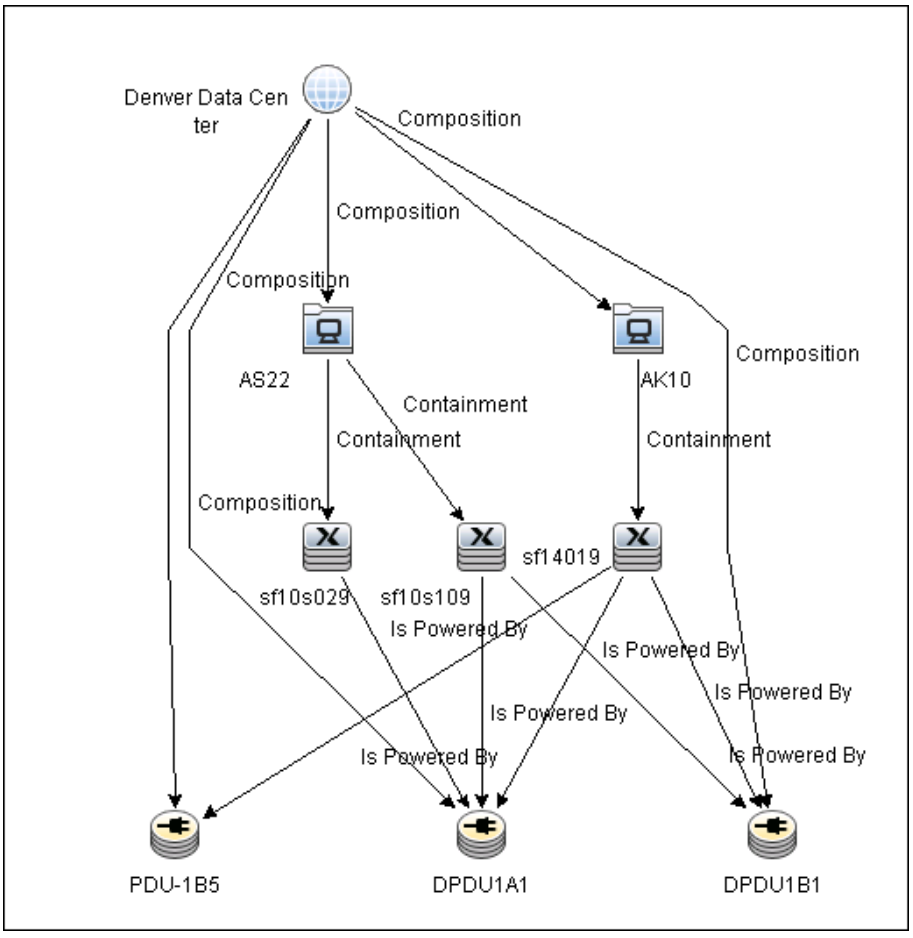
Integration is accomplished by running SQL queries on the Aperture VISTA SQL database.

Supported Versions

UCMDB supports integration with Aperture VISTA version 600. Aperture Integration Management Software Package version 2.0 or later is required.

Topology

The following image displays datacenter topology from VISTA.



Note: For a list of discovered CITs, see "Discovered CITs" on page 466.

How to Use the Aperture VISTA Integration Adapter

1. **Prerequisite - Scripts**

   The following scripts should be run on the VISTA database:

   - **v600_VIP_DAL_DV_Devices.sql**
   - **v600_Device_to_PDU.sql**

     The scripts are located in the discovery probe at **<hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\VistaScripts**

2. **Prerequisite - Credentials**

   Population is accomplished using SQL queries over JDBC. The following credentials should be defined:

   - Generic DB Protocol (SQL)

   For credential information, see "Supported Protocols" on page 82.

3. **Run the job**

   a. Run **Range IPs by ICMP** to discover the IP address of the SQL Server used by Aperture VISTA.

   b. Run **Database TCP Ports** to discover SQL Server ports on the IP addresses discovered above.

   c. Run **MSSQL Connection by SQL** to discover the SQL Server instance used by Aperture VISTA.

   d. Run **MSSQL Topology by SQL** to discover database instances in the SQL Server instance discovered above.

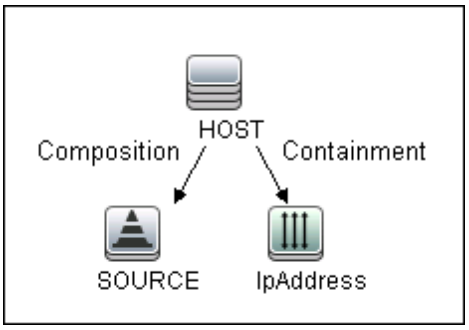
   e. Create a new integration point, and use the **Aperture VISTA by SQL** adapter to discover datacenter and power infrastructure from VISTA.

Aperture VISTA by SQL Adapter

Input CIT

Microsoft SQL Server

Input Query



Triggered CI Data

Name	Value
credentialsId	${SOURCE.credentials_id}
ip_address	${SOURCE.application_ip}
port	${PROCESS.application_port}

Used Scripts

Aperture_Vista_by_SQL.py

Discovered CITs

- Chassis
- Composition
- Containment
- Datacenter
- DatacenterResource
- Node
- PowerDistributionUnit
- Rack
- RemotePowerPanel
- Usage

Discovery Mechanism

Aperture VISTA uses a Microsoft SQL Server database as its data store. The Aperture Integration Management software package (v2.0 or greater) adds some views to the VISTA database, and this integration adapter collects information by running SQL Queries against these views.

The integration adapter in this package is triggered by SQL Server instances in UCMDB that have a database instance with **vista** in their name. Out-of-the-box discovery jobs may be used to discover these SQL Server database instances.

The adapter works as follows:

1. **Datacenter and Power Infrastructure Details**

   It runs the following SQL query to get details on Datacenter and Power infrastructure from the Aperture VISTA data store:

   ```
   SELECT device_name, device_serial_number, device_asset_class,
   device_manufacturer, device_model, device_model_info, rack_name,
   rack_asset_number, rack_serial_number, row_name, grid_location,
   space_name, floor, building_name, parent_name, parent_serial_number

   FROM vista.dbo.vip_dal_dv_devices

   WHERE device_asset_class='SERVER' OR device_asset_class='PDU' OR
   device_asset_class='RPP' OR device_asset_class='CHASSIS' OR device_
   asset_class='RACK'

   GROUP BY device_name, device_serial_number, device_asset_class,
   device_manufacturer, device_model, device_model_info, rack_name,
   rack_asset_number, rack_serial_number, row_name, grid_location,
   space_name, floor, building_name, parent_name, parent_serial_number

   ORDER BY device_asset_class, device_name
   ```

2. **Identification of Power Supply Routing**

   After all the infrastructure, power, and server CIs are created, the adapter uses the following query to identify power supply routing to servers:

   ```
   SELECT downstream_device_name, downstream_device_serial_number,
   upstream_device_name, upstream_building_name

   FROM vista.dbo.vip_dal_pwr_device_power_sources

   WHERE (downstream_device_name IS NOT NULL OR downstream_device_
   serial_number IS NOT NULL) AND upstream_device_name IS NOT NULL AND
   upstream_building_name IS NOT NULL AND upstream_node_type='PDU'

   GROUP BY downstream_device_name, downstream_device_serial_number,
   upstream_device_name, upstream_building_name

   ORDER BY downstream_device_name
   ```

Chapter 38

HP Asset Manager Integration

This chapter includes:

Overview

HP Asset Manager is an asset lifecycle management solution with modular components allowing an IT organization to measure and communicate the value it provides to the business it supports.

Supported Versions

The Asset Manager adapter supports Asset Manager Versions 5.22 and later.

Integration Overview

UCMDB-Asset Manager integration is implemented by the Asset Manager adapter (AMAdapter) pulling CIs and relationships from Asset Manager to UCMDB.

How to Integrate Asset Manager with UCMDB

This task consists of the following steps:

1. **Prerequisites - Deploying the Asset Manager package**

   Ensure the following steps are completed as detailed in "Integrating Asset Manager with HP Universal CMDB" in the *HP SACM Solution Configuration Guide*, available at: <hp>\UCMDB\UCMDBServer\runtime\fcmdb\CodeBase\AMAdapter\SACM9.02.U1-Configuration-EN.pdf.

   Note: You can also access it from the **AMAdapter** package, in the path: **adapterCode\AMAdapter\SACM9.02.U1-Configuration-EN.pdf**. You can access this package from the Package Manager (**Administration > Package Manager**). And you can also access the document by exporting the package to a zip file.

   - **Create the Asset Manager SQL views**
   - **Deploying the Asset Manager integration package to HP Universal CMDB**
   - **Making some CI attributes visible**
   - **Mapping the location types in Asset Manager and HP Universal CMDB**
   - **Adding the asset_tag attribute**

2. **Create the integration point**

   In DFM, in the Integration Studio, create a new integration point.

   a. Provide a name and description for the integration point.

   b. Ensure the **Is Integration Activated** option is enabled.

   c. Under **Integration Properties > Adapter**, select **HP Software Products > Asset Manager >**:

      Asset Manager Adapter 9.02 Update 1 for SACM 9.02 Update 1

      Asset Manager Adapter for SACM 9.02 or earlier

d. Under Adapter Properties:

Field	Value
Hostname\IP	<host name or IP address of computer hosting the Asset Manager database>\<name of the instance used by the database>
Port	Type the port to access the Asset Manager database.
Credentials ID	See **Create credentials**, below.
DB Name/SID	Type the database identifier used by Asset Manager.
DB Type	Select the database type. For example: SQL Server.
Probe Name	Select an appropriate probe.

e. Create credentials

i. Click the ellipsis to the right of the **Credentials ID** property.

ii. Select the **Generic DB Protocol (SQL)**protocol in the left pane.

iii. Click **add new connection details for selected protocol type**.

iv. Populate the fields on the **SQL Protocol Parameters** page, **General** section, as follows:

Field	Value
Network Scope	Use the default value.
User Label	Type a label for the credential.

v. Populate the fields on the **SQL Protocol Parameters** page, **SQL** section, as follows:

Field	Value
Database Type	Select the DBMS type.
Port Number	The port to access the database.
Connection Timeout	The time in milliseconds after which the probe stops trying to connect to the database.
User Name	The name of the user used to connect to the database.
Password	The password of the user needed to connect to the database.

Note: For details about integration points and credentials, see the *HP Universal CMDB Data Flow Management Guide*.

f. Click **Test Connection** to verify the connection is successfully established.

g. Click OK.

h. Save the integration point.

3. **Run the appropriate integration data flow:**
   - Population
   - Federation

> **Note:** For details on running these, see "Integrating Asset Manager with HP Universal CMDB" in the *HP SACM Solution Configuration Guide.*

Adapter

This job uses the adapter called AMAdapter.

Input CIT

destination_config

Triggered CI Data

Name	Value
adapterId	${ADAPTER.adapter_id}
attributeValues	${SOURCE.attribute_values}
credentialsId	${SOURCE.credentials_id}
destinationId	${SOURCE.destination_id}

Parameters

Name	Value
dbname	AssetManager
dbtype	SQLServer
port	1433

Troubleshooting and Limitations

When running a diff population job, ensure **amComputer.dtLastModif** for related CIs is updated.

For more information, see the *HP SACM Solution Configuration Guide.*

Chapter 39

Atrium Integration

This chapter includes:

Overview

UCMDB-Atrium integration consists of two independent, bi-directional parts: the **Data Push into Atrium** and the **Population from Atrium**.

- The **Data Push into Atrium** in UCMDB replicates CIs and relationships to Atrium and Remedy.

  The out-of-the-box integration does not transfer a specific list of CIs and relationships, but does enable you to replicate any CI or relationship from UCMDB to Remedy or Atrium.

  For examples of enabling the integration with commonly used CIs and relationships, see "Configure synchronization queries" on page 477.

- The **Population from Atrium** in UCMDB pulls CIs and relationships from Atrium to UCMDB.

Supported Versions

HP Universal CMDB integrates with the following BMC products:

- BMC Remedy Service Desk (Remedy) versions 7.0, 7.1, 7.5, 7.6
- BMC Atrium CMDB (Atrium) versions 2.0, 2.1, 7.5, 7.6

How to Work with the Data Push into Atrium Adapter

This task includes the following steps:

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

   Make sure that you have set up the Remedy protocol.

   For credential information, see "Supported Protocols" on page 82.

2. **Configure the Properties file**

Configure the **push.properties** file: **Data Flow Management > Adapter Management > Resources > Packages > AtriumPushAdapter > Configuration Files > push.properties**.

Property	Description
jythonScript.name	The name of the Jython script that is invoked by this push adapter.
mappingFile.default	The default XML mapping file used by mapping if a specific XML mapping file is not defined for an integration query. At least one default mapping file must be present in every adapter.
DebugMode	If this value is set to **true**, the CI and relationships being pushed to Remedy/Atrium are also saved to XML files on the Data Flow Probe, under the following folder: **/discoveryResource/AtriumPushAdapter/work**.
smartUpdateIgnoreFields	A comma separated list of attributes (transferred from UCMDB to Atrium) that should **not** be used to check whether a CI has changed in Atrium. For example, as **updateTime** always changes, you would not want to update a CI in Atrium just because this attribute has changed.
sortCSVFields	Parameter that includes the TQL results of CSV aggregated fields that must always be sorted. When child attribute values are mapped and aggregated as CSV, the results are not sorted. This can trigger an update, even though nothing has changed in Atrium. To prevent an update, add here the CSV aggregated fields that must always be sorted.
testConnNameSpace	Must be set to the **BMC NameSpace** being used for test connection purposes (for example, **BMC.CORE**).
testConnClass	Must be set to the name of a BMC class, to query for connection test purposes (for example, **BMC_ ComputerSystem**).

3. ## Configure the Data Flow Probe

- Copy the following JAR and DLL files from the BMC server to the following directory on the Data Flow Probe Server: **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\ AtriumPushAdapter**.

  This directory is automatically created once the **AtriumPushAdapter** package is deployed on the UCMDB Server. If it is not present, ensure that the **AtriumPushAdapter** package has been correctly deployed on the UCMDB Server.

  For details on deploying packages, see "Package Manager" in the *HP Universal CMDB Administration Guide*.

JAR Files	DLL Files
arapi75.jar	arapi75.dll
arutil75.jar	arencrypt75.dll
cmdbapi75.jar	arjni75.dll
commons-beanutils.jar	arrpc75.dll
commons-codec-1.3.jar	arutiljni75.dll
commons-collections-3.2.jar	arutl75.dll
commons-configuration-1.3.jar	arxmlutil75.dll
commons-digester-1.7.jar	cmdbapi75.dll
commons-lang-2.2.jar	cmdbjni75.dll
log4j-1.2.14.jar	icudt32.dll
oncrpc.jar	icuinbmc32.dll
spring.jar	icuucbmc32.dll
	Xalan-Cbmc_1_9.dll
	XalanMessagesbmc_1_9.DLL
	xerces-cbmc_2_6.dll
	xerces-depdombmc_2_6.dll

Note:

- The AR System Java API is forward and backward compatible with other versions of the AR System. For a complete compatibility matrix, refer to the "API Compatibility" section in the *BMC Remedy/Atrium Developer Reference Guide*.
- The arencrypt*.dll files are only required if encryption is enabled on the Remedy server.

a. Edit the **WrapperGateway.conf** file (or **WrapperManager.conf** if the Probe Manager and Gateway are running in separate mode) in the following directory: **C:\hp\UCMDB\DataFlowProbe\bin**.

Add the following line after the **wrapper.java.library.path.2=%content_dll%** line:

```
wrapper.java.library.path.3=%runtime%/probeManager
/discoveryResources/AtriumPushAdapter
```

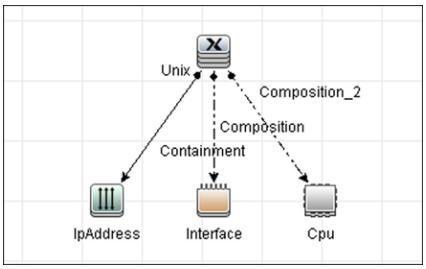
b. Add the complete path to the Atrium DLL files (for example, **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\AtriumPushAdapter**) to the Windows System Path on the Data Flow Probe machine.

c. Restart the Data Flow Probe service.

4. ## Configure synchronization queries

   The CIs and relationships to be pushed to Remedy/Atrium must be queried from UCMDB. Create queries (of type **Integration**) to query the CIs and relationships that have to be pushed to Remedy/Atrium.

   An example of such a query (**atrium_push_sample_query**) is included with the Atrium package. To access the query, navigate to **Modeling > Modeling Studio > Root > Integration > Atrium**.

5. ## Create XML mapping files

   For every query created in the step above, create an XML mapping file with the same name as the integration query (the name must have the same case) in the following directory:

   C:\hp\UCMDB\UCMDBServer\runtime\fcmdb\CodeBase\AtriumPushAdapter\mappings

   A sample mapping file (**atrium_push_sample_query.xml**) is provided out-of-the-box with the Atrium package.

   For more details, see "Mapping Files" on page 485.

6. ## Create an integration point

   For details about creating an integration point, see "Integration Point Pane" in the *HP Universal CMDB Data Flow Management Guide*.

   a. In the Integration Studio, create an integration point, selecting the **Data Push into Atrium** adapter. Enter the following information:

Name	Description
Credentials	○ Select **Remedy Protocol**. ○ Select the credentials to be used with this integration point. For credential information, see "Supported Protocols" on page 82.
Hostname/IP	The host name or IP address of the BMC Remedy server.
Integration Name	The name you give to the integration point.

Name	Description
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Port	The port number of the BMC Remedy server.
Probe Name	Select the Probe that should run this integration.

b. Test the connection. If a connection is not successfully created, check the integration point parameters and try again.

c. Save the integration point**.**

7. ## Define a Job

For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

Select the queries that will synchronize data between UCMDB and Remedy/Atrium. Save the job definition and the integration point.

8. ## Invoke a full run of the job

In the Integration Studio, on the Job Definition tool bar, click  to run a full discovery job. For details, see "Integration Jobs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

How to Work with the Population from Atrium Adapter

This task includes the following steps:

1. **Prerequisites - File preparation**

   a. Get the following files from the Remedy ARS and Atrium system. Note that all are required.

JAR Files	DLL Files
arapi75.jar	arapi75.dll
arutil75.jar	arencrypt75.dll
cmdbapi75.jar	arjni75.dll
commons-beanutils.jar	arrpc75.dll
commons-codec-1.3.jar	arutiljni75.dll
commons-collections-3.2.jar	arutl75.dll
commons-configuration-1.3.jar	arxmlutil75.dll
commons-digester-1.7.jar	cmdbapi75.dll
commons-lang-2.2.jar	cmdbjni75.dll
log4j-1.2.14.jar	icudt32.dll
oncrpc.jar	icuinbmc32.dll
spring.jar	icuucbmc32.dll
	Xalan-Cbmc_1_9.dll
	XalanMessagesbmc_1_9.DLL
	xerces-cbmc_2_6.dll
	xerces-depdombmc_2_6.dll

   Note:

   - The AR System Java API is forward and backward compatible with other versions of the AR System. For a complete compatibility matrix, refer to the "API Compatibility" section in the *BMC Remedy/Atrium Developer Reference Guide*.
   - The arencrypt*.dll files are only required if encryption is enabled on the Remedy server.

   b. Edit the **WrapperGateway.conf** file (or **WrapperManager.conf** if the Probe Manager and Gateway are running in separate mode) in the following directory: **C:\hp\UCMDB\DataFlowProbe\bin**.

   Add the following line after the **wrapper.java.library.path.2=%content_dll%** line:

```
wrapper.java.library.path.3=%runtime%/probeManager
/discoveryResources/AtriumPushAdapter
```

c. Add the complete path to the Atrium DLL files (for example, **C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\AtriumImportAdapter**) to the Windows System Path on the Data Flow Probe machine.

d. Restart the Data Flow Probe service.

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

Configure a generic protocol with the ARS server's username and password.

> **Note:** While creating the generic protocol, set the protocol description to **atrium**.

3. **Prerequisites - Create XML mapping files**

This step involves creating XML mapping files (in the **<probe>\runtime\probeManager\discoveryResources\TQLExport\Atrium\data** directory). These files map the BMC Atrium classes, attributes and relationships to their UCMDB equivalents. To create the XML mapping files for the topology requires identification of the topology to be imported from Atrium, and ensuring an equivalent topology exists in UCMDB. For more details, see "Mapping Files" on page 485.

4. **Run the job - UCMDB 9.04 and later**

In DFM, in the Integration Studio, create a new integration point.

a. Provide a name and description for the integration point.

b. Under **Integration Properties > Adapter**, select the **Population from Atrium** adapter.

c. Configure the following adapter properties:

i. **ARS_Server**

ii. **ARS_Port**

iii. **BMC_NameSpace**

d. Under **Adapter Properties > Probe Name** select the **Data Flow Probe** which will be used for the integration.

e. Under **Adapter Properties > Trigger CI instance** select:

i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI, or

ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

> **Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

f. Save the integration point.

g. Run the job.

> **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

5. **Run the job - UCMDB 9.03 and 9.02**
   a. Configure the following attributes for the job Import data from Atrium:
      i. **ARS_Server**
      ii. **ARS_Port**
      iii. **BMC_NameSpace**
   b. Run the **Import data from Atrium** job.

Atrium Push Job

Adapter

This discovery uses the adapter called **Data Push into Atrium**.

Used Scripts

pushToAtrium.py

Parameters

Parameter	Description
credentialsId	The credentials ID to use for Atrium connection.
host	The host name or IP address of the remote Atrium server
port	The Atrium server's connection port (if not using portmapper)
probeName	An internal setting which UCMDB will automatically replace.

Integration Flow

Integration includes the following activities:

1. **Querying the UCMDB for CIs and relationships**. When an ad-hoc integration job is run in the Integration Studio, the integration process:
   a. Receives the names of the integration queries that are defined in the job definition for that integration point.
   b. Queries UCMDB for the results (new, updated, or deleted CIs and relationships) of these defined queries.
   c. Applies the mapping transformation according to the pre-defined XML mapping files for every query.
   d. Pushes the data to the Data Flow Probe.
2. **Sending the data to BMC Remedy/Atrium**. On the Data Flow Probe, the integration process:
   a. Receives the CI and relationship data sent from the UCMDB Server.
   b. Connects to the BMC Remedy/Atrium server using the Java API.
   c. Transfers the CIs and relationships.

Import Data from Atrium Job

Adapter

This discovery uses the adapter called **Population from Atrium**.

Input CIT

The input CIT for this adapter is - **discoveryprobegateway**. The job uses an instance of the Discovery Probe Gateway which has access to connect to the remote BMC Atrium server.

Used Scripts

The adapter uses the following scripts:

Script	Description
atrium_query.py	Used to query BMC Atrium for data.
atrium_map.py	Used to map the queried data into data UCMDB can use.
atrium_to_ucmdb.py	Used to push imported data into UCMDB.

Discovered CITs

This integration can discover any CIT or relationship which is (a) mapped in the integration and (b) can be queried and converted to its UCMDB equivalent.

Parameters

Parameter	Detail
ARS_Port	The port for connecting to the ARS server. If portmapper is being used, this should be left as 0. Otherwise, specify the TCP port.
ARS_Server	The hostname or IP address of the BMC ARS server.
BMC_NameSpace	The BMC NameSpace to use. (For example: BMC.CORE.)
ChunkSize	The chunk size in which data should be retrieved from the remote server.
DateParsePattern	Set the date pattern to parse Atrium date strings.
DebugMode	Set to **true** to run integration in debug mode; this does not send data to UCMDB

Integration Flow

The Population from Atrium integration adapter flow has the following steps:

1. **Querying the Atrium server**

   In this step, the integration adapter connects to the Atrium server and queries it for classes, attributes and relationships, described in the XML mapping files. The result of this step is the creation of intermediate XML files (in the **<probe>\runtime\probeManager\discoveryResources\TQLExport\Atrium\inter** directory).

2. **Mapping the data**

   In this step, the data collected from the previous step and stored in the intermediate XML file, is converted into the UCMDB data format based on the mappings defined in the XML mapping files.

3. **Pushing the data to the UCMDB server**

   In this final step, after being mapped into the UCMDB object state holder vector format, the data is sent to the UCMDB server.

Mapping Files

This section includes:

Mapping Files Overview

A mapping file is an XML file that defines which CIT or relationship in UCMDB is mapped to which CIT or relationship in the target data store.

Mapping files:

- Control which CITs and relationships are to be pushed.
- Control the attributes for the CITs and relationships that are to be mapped.
- Map attribute values from multiple CIs to one target CI.
- Map attributes of children CIs (those having a **containment** or **composition** relationship) to the parent CI in the target data store. For example:
  - Set a **Number of CPUs** value for a target **node** CI.
  - Set a **Total Memory** value for a target **node** CI.
- Map attributes of parent CIs (those having a **containment** or **composition** relationship) in the target data store CI. For example, in the Atrium target data store, set the value of a **Container Server** attribute on the **Installed Software** CIT by retrieving the value of the UCMDB **Installed Software** CI container node.

Mapping File Structure

Every mapping file has the following skeletal structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<integration>
    <info>
        <source ... ... />
        <target ... ... />
    </info>
    <source_ci_type name="...">
        <target_ci_type name="...">
            <targetprimarykey>
                <pkey>...</pkey>
            </targetprimarykey>
            <target_attribute name="..." datatype="..." >
                <map type="..." />
            </target_attribute>
        </target_ci_type>
    </source_ci_type>
</integration>
```

Note: An elipsis (**...**) signifies a configurable section.

Mapping File Elements

This section includes:

- Main Parent Elements
- CI Type Mapping Elements
- Relationship Type Mapping Elements

Main Parent Elements

- **<integration>**. The root element of the XML file. This element has no attributes.
- **<info>**. The source and target data stores being used, for example:

```
<info>
<source name="Atrium"  versions="7.6" vendor="BMC" />
<target name="UCMDB" versions="9.0" vendor="HP" />
</info>
```

- **<targetcis>**. The element that encapsulates the mapping for all CI types.
- **<targetrelations>**. The element that encapsulates the mapping for all relationship types.

CI Type Mapping Elements

- **<source_ci_type>**. The element that defines a CI type of the source data store, for example:

```
<source_ci_type name="BMC_ComputerSystem" nameSpace="BMC.CORE"
query="">
```

  - **Attribute: name**. Defines the name of the source CI type.
  - **Attribute: mode**. Defines the mode of the update in the target data store.
- **<target_ci_type>**. The element that defines the target CIT, for example:

```
<target_ci_type name="unix">
```

  - **Attribute: name**. Defines the name of the target CIT.
- **<targetprimarykey>**. The element that defines a list of all primary keys of the target CIT, for example:

```
<targetprimarykey>
    <pkey>host_key</pkey>
</targetprimarykey>
```

- **<target_attribute>**. The element that defines an attribute mapping from the source CI type to the target CI type attribute. Attribute mapping can be of the following types:
  - **Constant**. This type enables setting a constant value on the target attribute:

```
<target_attribute name="data_note" datatype="string" length="127">
<map type="constant" value="ATRIUM DATA" />
</target_attribute>
```

  - **Direct**. This type enables setting a direct value of a source data store attribute on the target data store:

```
<target_attribute name="name" datatype="string">
<map type="direct" source_attribute="Name" />
</target_attribute>
```

  - **Compound String**. This type enables the use of the above mapping types together to form more complex values for the target attribute, for example:

```
<target_attribute name="Bunch_O_Data" datatype="char" length="510"
option="uppercase">
<map type="compoundstring">
    <source_attribute name="name"/>
    <constant value="_UNIX_Server, IP="/>
    <childattr name="ip_address" source_attribute="ip_address"
aggregation="csv"/>
    <constant value=", CPU="/>
    <childattr name="cpu" source_attribute="display_label"
aggregation="csv"/>
</map>
</target_attribute>
```

Relationship Type Mapping Elements

- **<link>**. The element that defines a relationship mapping from the source data store to a target data store, for example:

```
<link source_link_type="composition"
    target_link_type="BMC_HostedSystemComponents"
    source_ci_type_end1="unix"
    source_ci_type_end2="cpu"
    role1="Source"
    role2="Destination    "
    mode="update_else_insert">
       <target_ci_type_end1 name="BMC_ComputerSystem"
superclass="BMC_System" />
       <target_ci_type_end2 name="BMC_Processor"
superclass="BMC_SystemComponent" />
... Relationship attribute mapping elements similar to the CI type
attribute mapping elements ...
</link>
```

  - **Attribute: source_link_type**. Defines the name of the source link.
  - **Attribute: target_link_type**. Define the name of the target link.
  - **Attribute: source_ci_type_end1**. The **End1** CI type of the source link.
  - **Attribute: source_ci_type_end2**. The **End2** CI type of the source link.
  - **<target_ci_type_end1>**. Used to specific the value of the target links end1 CI type
  - **<target_ci_type_end2>**. Used to specific the value of the target links end2 CI type

Troubleshooting and Limitations

The integration mapping file enables the mapping only of concrete CI types and relationships to the CI types and relationships in BMC Remedy/Atrium. That is, a parent CIT cannot be used to map children CIs. For example, if **UCMDB Node** is mapped to **BMC_ComputerSystem**, any Node CIT of type **Unix** is not transferred. A mapping must be separately created for **Unix** to **BMC_ ComputerSystem**.

Chapter 40

CA CMDB Integration

This chapter includes:

Overview

The UCMDB - CA CMDB integration adapter allows pushing CIs and relationships from UCMDB into CA CMDB.

This is achieved by querying the UCMDB for CIs and Relationships based on queries defined in the push integration adapter. The output of the queried CIs and Relationships are saved in an XML file.

GRLoader, a utility provided with CA CMDB, transfers the CIs and Relationship data stored in the XML file into CA CMDB. An XML mapping file is used to define how the CIs and Relationships in UCMDB are related to the CIs and Relationships in CA CMDB.

The CA CMDB integration package is bundled in **CACmdbPushAdapter.zip.**

Supported Versions

UCMDB supports integration with CA CMDB R12.5 and R12.0.

Integration Mechanism

This section describes the UCMDB - CA CMDB integration mechanism:

1. **UCMDB is queried for CIs and Relationships**

   When an ad-hoc job is run from the defined integration point, the integration receives the names of the integration queries that have been defined in the job definition for that integration point.

   The integration process queries UCMDB for the results of these queries (new/updated/deleted CIs and Relationships), and applies the mapping transformation according to the pre-defined XML mapping files for every query.

   It then pushes the data to the Data Flow Probes.

2. **Queried data is converted into temporary XML files on the Data Flow Probe system**

   On the Data Flow Probe side, the integration process receives the CI and Relationship data sent from the UCMDB server, and converts it into a format which can be used as input XML for the GRLoader, a utility provided with CA CMDB used to transfer the CI and Relationship data into CA CMDB.

3. **CA CMDB GRLoader utility is invoked on the Data Flow Probe system**

   Finally, the integration process programmatically invokes the CA CMDB GRLoader utility on the Data Flow Probe system with the necessary parameters (for example, CA CMDB server, port, username, and password), using the input XML file created in the previous step to transfers the CIs and Relationship data into CA CMDB.

How to Work with the CA CMDB Push Adapter

The CA CMDB push adapter allows replication of CIs and Relationships from UCMDB to CA CMDB.

This task includes:

1. ## Prerequisite - Set up the CA CMDB protocol

   This integration uses the **CA CMDB protocol**. For credential information, see "Supported Protocols" on page 82.

2. ## Prerequisite - Other

   - **Data Flow Probe System:**
     - Copy all of the files in the CA CMDB system's **%NX_ROOT%\java\lib** directory to the **CaCmdbPushAdapter** directory on the data flow probe system:

       ```
       <UCMDB Installation>\DataFlowProbe\runtime\probeManager\
       discoveryResources\CaCmdbPushAdapter
       ```

     - Locate the file, **NX.ENV,** in the **CaCmdbPushAdapter** directory. If the file does not exist, create it in the **CaCmdbPushAdapter** directory and add the following text to it:

       ```
       @NX_LOG=C:/CA/java/lib/log
       ```

     - Open **<UCMDB Installation>\DataFlowProbe\runtime\probeManager\discoveryConfigFiles\globalSettings.xml**, locate the following line, and add **",CaCmdbPushAdapter/*.*"** as illustrated in bold:

       ```
       db/or-
       acle/*.*;db/mssqlserver/*.*;db-
       /db2/*.*;db/sybase/*.*;nnm/*.*;AtriumPushAdapter/*.*
       ;CaCmdbPushAdapter/*.*
       ```

     - Restart the Data Flow Probe service.

3. ## Configure integration queries

   Create integration queries to query the CIs and Relationships that must be pushed from UCMDB to CA CMDB.

   For an example of such an integration query, see "Integration Query" on next page.

4. ## Create the XML mapping files

   For every integration query that you create, create an XML mapping file with the exact same name as the integration query (case-sensitive). Create the XML files in the following directory:

   ```
   <UCMDB Installation>\UCMDBServer\runtime\fcmdb\CodeBase\
   CaCmdbPushAdapter\mappings
   ```

   For more information about mapping files, see "Prepare the Mapping Files" in the *HP Universal CMDB Developer Reference Guide*.

Note: A sample mapping file, **Unix_SW_to_CACMDB.xml**, is provided out- of-the-box with the integration package.

5. **Create an integration point**

In UCMDB create an integration point. (For details, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.)

Include the following details:

a. Provide a name and description for the integration point.

b. Provide the following details for the **CaCmdbPushAdapter** adapter:

Attribute	Description
Hostname/IP	The host name or IP address of the CA CMDB server.
Port	The port number of the CA CMDB server.
Credentials	The CA CMDB credential that was created in the prerequisites section above
Probe Name	The name of the Data Flow Probe on which the integration will run.

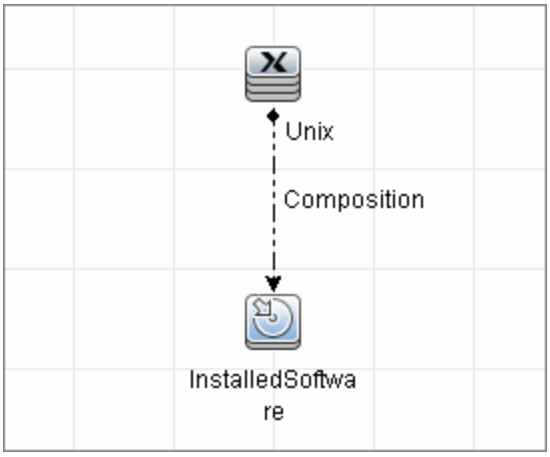
c. Test the connection to the target CMDB server.

d. Add a job definition to the integration point, selecting the queries to use to synchronize data between UCMDB and CA CMDB. Define a synchronization schedule, if required.

e. Invoke the ad hoc job, **Full Topology Sync**, for a full synchronization of the data.

Integration Query

The integration query, **Unix_SW_to_CACMDB**, is included with CA CMDB integration package. This is an example of a query that can be used to query the CIs and Relationships that must be pushed from UCMDB to CA CMDB. This query is accessible from UCMDB's Modeling Studio, among the query resources. For details, see "Modeling Studio Page" in the HP Universal CMDB Modeling Guide.



Troubleshooting and Limitations

This section describes troubleshooting and limitations related to UCMDB - CA CMDB integration.

- **Debug Mode**

  To create an XML dump of the CIs and links being sent to the CA CMDB server for debug purposes, in **<UCMDB installation>\DataFlowProbe\runtime\probeManager\discoveryConfigFiles\CaCmdbPushAdapter\push.properties**, set the value of the **debugMode** property to **true** and restart the Data Flow Probe service.

  This ensures that every time the integration is invoked, a set of XML files is created in the **<UCMDB installation>\DataFlowProbe\runtime\probeManager\discoveryResources\CaCmdbPushAdapter\work** directory. These files are time-stamped and contain the CIs and links that UCMDB is trying to push to CA CMDB. This information can be helpful in debugging a problem with the integration:

  - If data is not being sent from UCMDB, there is a problem on the UCMDB side.
  - If data is not being processed by CA CMDB's GRLoader utility, there might be a reconciliation issue or some other issue on the CA CMDB side.

Chapter 41

CiscoWorks LAN Management Solution Integration

This chapter includes:

Overview

CiscoWorks LAN Management Solution (LMS) is a suite of management tools that simplify the configuration, administration, monitoring, and troubleshooting of Cisco networks.

This integration involves synchronizing devices, topology, and hierarchy of network infrastructure in UCMDB, and also synchronizes relationships between various hardware and logical network entities to enable end-to-end mapping of the data network infrastructure. The integration enables change management and impact analysis across all business services mapped in UCMDB, from a data network point of view.

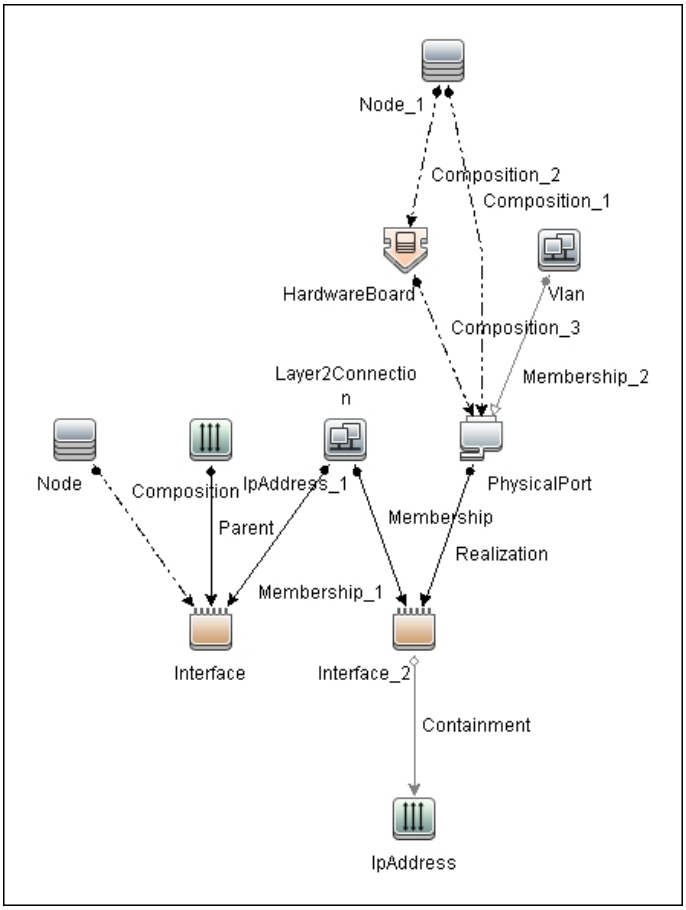
Supported Versions

This integration supports CiscoWorks LAN Management Solution Version 3.x.

Topology

The following image displays the CiscoWorks LAN Management Solution topology.

Note: For a list of discovered CITs, see "Discovered CITs" on page 502.



How to Discover CiscoWorks LMS

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

   Add credentials for the Sybase database instances used by CiscoWorks LMS (**RMENGDB** and **ANIDB**) to the Generic DB (SQL) protocol.

   For credential information, see "Supported Protocols" on page 82.

2. **Run the discovery**

   a. Discover IP addresses of the Sybase databases **RMENGDB** and **ANIDB** used by CiscoWorks LMS.

   b. Run the **CiscoWorks LMS Database Ports** job to discover the TCP ports at which the Sybase databases used by CiscoWorks LMS are listening.

   c. Create a new integration point, and use the **CiscoWorks NetDevices** adapter to discover network device information from CiscoWorks.

   d. Create a new integration point, and use the **CiscoWorks Layer 2** adapter to discover node (server) information from CiscoWorks.

   Steps 2a and 2b are optional (although highly recommended - see note below) since CiscoWorks adapters are available in the Integration Studio, allowing manual creation of the necessary **IpAddress**, **Node** and **IpServiceEndpoint** CIs.

Note: The CiscoWorks Layer 2 job requires additional data about CIs created by the **CiscoWorks NetDevices** adapter and already in UCMDB. This information is provided by the Input Query, which contains CI Types (**NetDevice** and **PhysicalPort**) that provide this data in addition to CI Types required to identify the integration target (**IpServiceEndpoint**). For this reason, it is highly recommended to execute steps 2a and 2b. If steps 2a and 2b are not executed, creating the integration target CIs (while creating an integration point using the **CiscoWorks Layer 2** adapter) requires the creation of **Node** and **PhysicalPort** CIs.

CiscoWorks LMS Database Ports Job

Adapter

This job uses the TCP Ports Discovery adapter

Trigger Query

- **Trigger CI**: IpAddress
- **Trigger Query**:

- **CI attribute conditions:**

CI	Attribute Value
IpAddress	NOT IP Probe Name is null

Parameters

Ports: 43443, 43455

Network Devices from CiscoWorks LMS Job

Adapter

This job uses the CiscoWorks_NetDevices adapter

Trigger Query

- **Trigger CI**: IpServiceEndpoint
- **Trigger Query**: CiscoWorks RME DB Port

- **CI attribute conditions:**

CI	Attribute Value
IpServiceEndPoint	NetworkPortNumber Equal 43455

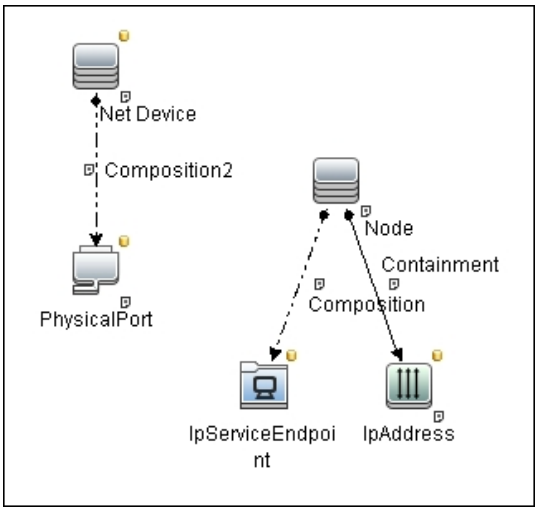
Layer 2 Topology from CiscoWorks LMS Job

Adapter

This job uses the CiscoWorks_Layer2 adapter

Trigger Query

- **Trigger CI**: IpServiceEndpoint
- **Trigger Query**: CiscoWorks Campus DB Port



- **CI attribute conditions:**

CI	Attribute Value
IpServiceEndPoint	NetworkPortNumber Equal 43443
IpAddress	NOT IP Probe Name Is null
NetDevice	NOT Name Is null
PhysicalPort	NOT Name Is null AND NOT Port VLAN Is null AND NOT PortIndex Is null AND NOT Container Is null

Discovery Flow

Add IP addresses of the Sybase databases **RMENGDB** and **ANIDB** used by CiscoWorks LMS to a discovery probe range:

1. Range IPs by ICMP
2. CiscoWorks LMS Database Ports

3. CiscoWorks NetDevices
4. CiscoWorks Layer 2

CiscoWorks NetDevices Adapter

Input CIT

IpServiceEndpoint: the TCP port at which the **RMENGDB** Sybase instance is listening. (The default is 43455.)

Used Scripts

- **ciscoworks_utils.py**
- **CiscoWorks_NetDevices.py**

Discovered CITs

- **Composition**
- **Containment**
- **HardwareBoard**
- **Interface**
- **IpAddress**
- **IpSubnet**
- **Layer2Connection**
- **Membership**
- **Node**
- **PhysicalPort**
- **Realization**
- **Vlan**

Parameters

Parameter	Description
allowDnsLookup	If an IP address is not available for a node, setting this to **true** enables DNS lookup using the node name. **Default**: false.
ignoreNodesWithoutIP	If set to **false**, CIs for nodes without IPs are created with the storage system's internal ID as **host_key**. **Note**: this may result in duplicate nodes. **Default**: true.
rmeDbName	The name of the CiscoWorks Resource Manager Essentials database in Sybase. **Default**: rmengdb.
queryChunkSize	The number of network devices to query at a time. **Default**: 250.

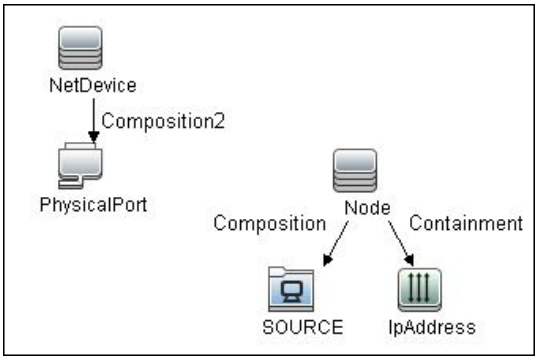
CiscoWorks Layer 2 Adapter

Input CIT

IpServiceEndpoint: the TCP port at which the **ANIDB** Sybase instance is listening. (The default is 43443.)

Input Query

CiscoWorks LMS Campus DB with PhysicalPorts



Node Conditions

Node Name	Condition
SOURCE (IpServiceEndPoint)	NetworkPortNumber Equal 43443
IpAddress	NOT IP Probe Name Is null
NetDevice	NOT Name Is null
PhysicalPort	NOT Name Is null AND NOT Port VLAN Is null AND NOT PortIndex Is null AND NOT Container Is null

Triggered CI Data

Name	Value
db_port	${SOURCE.network_port_number}
ip_address	${IpAddress.ip_address}
netdevice_cmdbid	${NetDevice.global_id}
netdevice_name	${NetDevice.name}
port_cmdbid	${PhysicalPort.global_id}
port_container_cmdbid	${PhysicalPort.root_container}
port_index	${PhysicalPort.port_index}
port_name	${PhysicalPort.name}
port_vlan	${PhysicalPort.port_vlan}

Used Scripts

- **ciscoworks_utils.py**
- **CiscoWorks_Layer2.py**

Discovered CITs

- **Composition**
- **Containment**
- **HardwareBoard**
- **Interface**
- **IpAddress**
- **IpSubnet**
- **Layer2Connection**
- **Membership**
- **Node**
- **PhysicalPort**
- **Realization**
- **Vlan**

Parameters

Parameter	Description
allowDnsLookup	If an IP address is not available for a node, setting this to **true** enables DNS lookup using the node name. **Default**: false.
ignoreNodesWithoutIP	If set to **false**, CIs for nodes without IPs are created with the storage system's internal ID as **host_key**. **Note**: this may result in duplicate nodes. **Default**: true.
campusDbName	The name of the CiscoWorks Campus database in Sybase. **Default**: anidb.
queryChunkSize	The number of nodes to query at a time. **Default**: 1,000.

Discovery Mechanism

The adapters in this package connect to the Sybase databases used by CiscoWorks LMS using JDBC, and run SQL queries to retrieve information. The Sybase database instances are used as part of the trigger for jobs in this package. This allows the jobs to be included in UCMDB's spiral discovery schedule.

The package includes two adapters:

- **CiscoWorks NetDevices**, and
- **CiscoWorks Layer 2**.

CiscoWorks NetDevices triggers off the CiscoWorks Resource Manager Essentials database, and retrieves network devices, VLAN and layer two infrastructure from it.

CiscoWorks Layer 2 triggers off the CiscoWorks Campus Manager database, and retrieves nodes (servers). It associates them with VLANs and layer two infrastructure retrieved by **CiscoWorks NetDevices**.

Database queries executed by this package on the CiscoWorks databases are as follows:

> **Note:** The following query is used by the **CiscoWorks NetDevices** and **CiscoWorks Layer 2** adapters on the **RMENGDB** and **ANIDB** database instances

Get the database name to verify that queries are run on the correct database:

```
SELECT db_name()
```

> **Note:** The following queries are used by the **CiscoWorks NetDevices** adapter on the **RMENGDB** database instance

Get a count of the number of network devices in the database (This is required to determine the number of chunks to query. For details on chunking, see "Parameters" on page 503.)

```
SELECT COUNT(1) FROM lmsdatagrp.NETWORK_DEVICES
```

Get information on network devices managed by CiscoWorks LMS

```
SELECT netdevices.Device_Id,
deviceState.NetworkElementID, netdevices.Device_Display_Name,
netdevices.Host_Name, netdevices.Device_Category,
netdevices.Device_Model, netdevices.Management_IPAddress,
deviceState.Global_State
FROM lmsdatagrp.NETWORK_DEVICES netdevices JOIN dba.DM_Dev_State
deviceState ON netdevices.Device_Id=deviceState.DCR_ID
```

Get additional details on each network device.

```
SELECT * FROM dba.PhysicalTypeEnum
```

```
SELECT ne.ElementName, ne.ReportedHostName, ne.DNSDomainName,
ne.Description, ne.PrimaryOwnerContact, ne.ElementLocation, os.OSName,
```

```
os.Version, os.ROMVersion, pe.Manufacturer, pe.SerialNumber
FROM dba.OperatingSystem os, dba.PhysicalElement pe,
dba.networkelement ne
WHERE os.NetworkElementID=<networkDeviceID> AND
ne.NetworkElementID=<networkDeviceID> AND
pe.NetworkElementID=<networkDeviceID> AND
LOWER(pe.PhysicalType)=<physicalType> AND pe.PhysicalElementId IN (1,
2)
```

Get port and VLAN information for each network device.

```
SELECT phyPort.PhysicalPortID, phyPort.SNMPPhysicalIndex,
phyPort.ParentRelPos, port.PORT_NAME, port.PORT_DESC, port.PORT_
DUPLEX_MODE, port.PORT_TYPE, port.PORT_SPEED, port.VLAN_NAME,
port.VLANID, interface.EndpointID, interface.Description,
interface.Alias, interface.MediaAccessAddress
FROM lmsdatagrp.PORT_INVENTORY port JOIN dba.PhysicalPort phyPort ON
port.PORT_NAME=phyPort.PortName JOIN dba.IFEntryEndpoint interface ON
port.PORT_NAME=interface.EndpointName
WHERE phyPort.NetworkElementID=<networkDeviceID> AND
interface.NetworkElementID=<networkDeviceID> AND port.DEVICE_
ID=<networkDeviceID> AND phyPort.PortName=port.PORT_NAME
```

Get IP Address details for each network device.

```
SELECT IPAddress, SubnetMask FROM dba.IPProtocolEndPoint WHERE
NetworkElementId=<networkDeviceID>
```

Get information on modules in each network device.

```
SELECT MODULE_NAME, SW_VERSION, FW_VERSION, SLOT_NUMBER FROM
lmsdatagrp.MODULE_INVENTORY WHERE DEVICE_ID=<networkDeviceID>
```

Note: The following queries are used by the **CiscoWorks Layer 2** adapter on the **ANIDB** database instance.

Get a count of the number of nodes (servers) in the database (This is required to determine if chunking is required. See "Parameters" on page 506.)

```
SELECT COUNT(1) FROM lmsdatagrp.End_Hosts
```

Get information on nodes managed by or known to CiscoWorks LMS.

```
SELECT HostName, DeviceName, Device, MACAddress, IPAddress,
SubnetMask, Port, PortName, VLAN, VlanId, associatedRouters
FROM lmsdatagrp.End_Hosts
WHERE HostName IS NOT NULL AND NOT HostName='' AND IPAddress IS NOT
NULL AND NOT IPAddress=''
```

Troubleshooting and Limitations

If there is a database connection failure, copy the Sybase JDBC driver (**jconnect*nn*.jar** or similar JAR file) from the Sybase system to the **<hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\db\sybase** directory on the DFM probe file system.

If the database connection failure occurs after the driver is copied, it may be necessary to change the driver classes in **globalSettings.xml** from:

<Sybase>com.sybase.jdbc.SybDriver</Sybase>

to

<Sybase>com.sybase.jdbc3.SybDriver</Sybase>

Chapter 42

Data Dependency and Mapping Inventory Integration

This chapter includes:

Overview

This document describes how to integrate DDMi with UCMDB. Integration occurs by populating the UCMDB database with devices, topology, and hierarchy from DDMi and by federation with DDMi's supported classes and attributes. This enables change management and impact analysis across all business services mapped in UCMDB.

According to UCMDB reconciliation rules, if a CI is mapped to another CI in the CMDB, it is updated during reconciliation; otherwise, it is added to the CMDB.

Supported Versions

DDMi integration has been developed and tested on HP Universal CMDB version 7.5.2 or later with ED version 2.20 or DDMi version 7.5.

DDMi Adapter

Integration with DDMi is performed using a DDMi adapter, which is based on the Generic DB Adapter. This adapter supports full and differential population for defined CI types as well as federation for other CI types or attributes.

The DDMi adapter supports the following features:

- Full population of all instances of the selected CI Types.
- Identifying changes that have occurred in DDMi, to update them in UCMDB.
- Implementing **Remove** in DDMi. When a CI is removed in DDMi, it is not physically deleted from the database, but its status is changed to indicate that the CI is no longer valid. The DDMi adapter interprets this status as an instruction to remove the CI when needed.
- Federation of defined CI Types and attributes.

Out-of-the-box integration with DDMi includes population of the following classes:

- Node (some of the attributes are populated and some are federated)
- Layer2 connection
- Location that is connected to the node
- IP address
- Interface

In addition, the following classes can be defined as federated from DDMi:

- Asset
- CPU
- File system
- Installed software

- Printer
- Cost center

The following classes and attributes should be marked as federated by the DDMi adapter for the proper functionality of the Actual State feature of Service Manager:

- Classes
  - Person
  - Asset
  - CPU
  - Installed software
  - Printer
  - Windows service
- Node attributes
  - DiscoveredOsVendor
  - DiscoveredModel
  - Description
  - DomainName
  - DiscoveredLocation
  - NetBiosName

> **Note:** Avoid marking the **CreateTime** and **LastModifiedTime** attributes as federated, as it may lead to unexpected results.

How to Populate the CMDB with Data from DDMi

This task describes how to install and use the DDMi adapter, and includes the following steps:

- "Define the DDMi integration" below
- "Define a population job (optional)" on next page
- "Run the population job" on next page

1. **Define the DDMi integration**
   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
   b. Click the **new integration point**  button to open the new integration point Dialog Box.

- Click  , select the DDMi adapter, and click **OK**.

  Each out-of-the-box adapter comes predefined with the basic setup needed to perform integration with UCMDB. For information about changing these settings, see "Integration Studio Page" in the *HP Universal CMDB Data Flow Management Guide*

- Enter the following information, and click **OK**:

Name	Description
Credentials	Allows you to set credentials for integration points. For credential information, see "Supported Protocols" on page 82.
Hostname/IP	The name of the DDMi server.
Integration Name	The name you give to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Port	The port through which you access the DDMi database.

c. Click **Test connection** to verify the connectivity, and click **OK**.

d. Click **Next** and verify that the following message is displayed: **A connection has been successfully created**. If it does not, check the integration point parameters and try again.

2. Define a population job (optional)

The DDMi adapter comes out-of-the-box with the DDMi Population job, which runs the following predefined queries: **hostDataImport**, **networkDataImport**, **printerDataImport**, and **Layer2DataImport**. For details about these queries, see "Predefined Queries for Population Jobs" on page 517. This job runs according to a default schedule setting.

You can also create additional jobs. To do this, select the Population tab to define a population job that uses the integration point you defined in "Define the DDMi integration" on previous page. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

3. Run the population job

Activate the population job in one of the following ways:

- To immediately run a full population job, click  . In a full population job, all appropriate data is transferred, without taking the last run of the population job into consideration.

- To immediately run a differential population job, click  . In a differential population job, the previous population time stamp is sent to DDMi, and DDMi returns changes from that time stamp to the present. These changes are then entered into the UCMDB database.

- To schedule a differential population job to run at a later time or periodically, define a

scheduled task. For details, see "Define Tasks that Are Activated on a Periodic Basis" in the *HP Universal CMDB Administration Guide*.

How to Federate Data with DDMi

The following steps describe how to define the CI Types that will be federated with DDMi.

1. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
2. Select the integration point that you defined in "Define the DDMi integration" on page 512.
3. Click the **Federation** tab. The panel shows the CI Types that are supported by the DDMi adapter.
4. Select the CI Types and attributes that you want to federate.
5. Click **Save**  .

How to Customize the Integration Data Model in UCMDB

Out-of-the-box CIs for DDMi integration can be extended in one of the following ways:

To add an attribute to an existing CI type:

If the attribute you want to add does not already exist in the CMDB, you need to add it. For details, see "Add/Edit Attribute Dialog Box" in the *HP Universal CMDB Modeling Guide*.

1. Navigate to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.
2. Locate the **generic_db_adapter.[CI type]** to be changed, and add the new attribute.
3. Ensure that the TQL queries that include this CI Type have the new attribute in their layouts, as follows:
   a. In the Modeling Studio, right-click the node where you want to include the attribute.
   b. Select **Query Node Properties**.
   c. Click **Advanced layout settings** and select the new attribute.

   For details about selecting attributes, see "Layout Settings Dialog Box" in the *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see "Troubleshooting and Limitations" on page 519

To add a new CI Type to the DDMi Adapter:

1. In UCMDB, create the CI Type that you want to add to the adapter, if it does not already exist. For details, see "Create a CI Type" in the *HP Universal CMDB Modeling Guide*.
2. Navigate to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.
3. Map the new CI type by adding a new entity called **generic_db_adapter.[CI type]**.
4. In the **orm.xml** file, ensure that the new CI Type has the following mappings:
   a. the **data_note** attribute is mapped to the **NMID_StatusInAppliance** column (this attribute is used for checking the CI's status).
   b. the **last_modified_time** and **create_time** attributes are mapped to the **Device_UpdatedDt** and **Device_FirstFoundDt** columns.

   For details, see "The orm.xml File" in the *HP Universal CMDB Developer Reference Guide*.
5. Create queries to support the new CI Types that you added. Make sure that all mapped attributes have been selected in the Advanced Layout settings:
   a. In the Modeling Studio, right-click the node where you want to include the attribute.
   b. Select **Query Node Properties**.
   c. Click **Advanced layout settings** and select the new attribute.

For details about selecting attributes, see "Layout Settings Dialog Box" in the *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see "Troubleshooting and Limitations" on page 519.

6. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
7. Edit the DDMi integration point to support the new CI Type by selecting it either for population or for federation.
8. If the new CI Type is for population, edit the population job that you created in "Define a population job (optional)" on page 513 to include the new TQL query.

Predefined Queries for Population Jobs

The following TQL queries (located in the Modeling Studio in the **Integration\Data In** folder) are provided out-of-the-box if you use the DDMi adapter when you create an integration point:

- **hostDataImport**- use to import nodes. Imported data includes nodes whose NodeRole attribute is either null, or contains **desktop**, **server**, or **virtualized_system**. Nodes are identified either by their interface or IP address. Information also includes the location of the nodes (building, floor and room).
- **networkDataImport** - use to import nodes that are not imported with **hostDataImport**. Similar to **hostDataImport**, except that it imports nodes whose NodeRole is not null and does not contain the following strings: **desktop**, **server**, **virtualized_system**, or **printer**.
- **printerDataImport** - use to import printers. Similar to **networkDataImport**, except that it does import nodes whose NodeRole contains the string **printer**.
- **Layer2DataImport** - use to import Layer2 connections between pairs of nodes through their interfaces. Information also includes the nodes and their IP addresses.

DDMi Adapter Configuration Files

The adapter includes the following configuration files:

- **orm.xml.** The Object Relational mapping file in which you map between UCMDB classes and database tables.
- **discriminator.properties.** Maps each supported CI type (also used as a discriminator value in **orm.xml**) to a list of possible corresponding values of the discriminator column, **DeviceCategory_ID**.
- **replication_config.txt.** Contains a comma-separated list of non-root CI and relations types that have a **Remove** status condition in the DDMi database. This status condition indicates that the device has been marked for deletion.
- **fixed_values.txt.** Includes a fixed value for the attribute **ip_domain** in the class IP (**DefaultDomain**).

For details on adapter configuration, see "Developing Generic Database Adapters" in the *HP Universal CMDB Developer Reference Guide*.

Troubleshooting and Limitations

Note: Only queries that meet these requirements are visible to the user when selecting a query for a population job.

- Queries that are used in population jobs should contain one CI Type that is labeled with a **Root** prefix, or one or more relations that are labeled with a **Root** prefix.

  The root node is the main CI that is synchronized; the other nodes are the contained CIs of the main CI. For example, when synchronizing the **Node** CI Type, that graph node is labeled as **Root** and the resources are not labeled **Root**.

- The TQL graph must not have cycles.
- A query that is used to synchronize relations should have the cardinality 1...* and an OR condition between the relations.
- The adapter does not support compound relations.
- The TQL graph should contain only CI types and relations that are supported by the DDMi adapter.
- ID conditions on the integration TQL query are not supported.

Chapter 43

EMC Control Center (ECC) Integration

This chapter includes:

Overview

Integration between ECC and DFM 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.

DFM initiates discovery on the ECC database. Synchronized Configuration Items (CIs) include Storage Arrays, Fibre Channel Switches, Hosts (Servers), Storage Fabrics, Storage Zones, Logical Volumes, Host Bus Adapters, Storage Controllers, and Fibre Channel Ports. The integration also synchronizes physical relationships between hardware, and logical relationships between Logical Volumes and hardware devices, to enable end-to-end mapping of the storage infrastructure.

You integrate ECC with UCMDB using Data Flow Management.

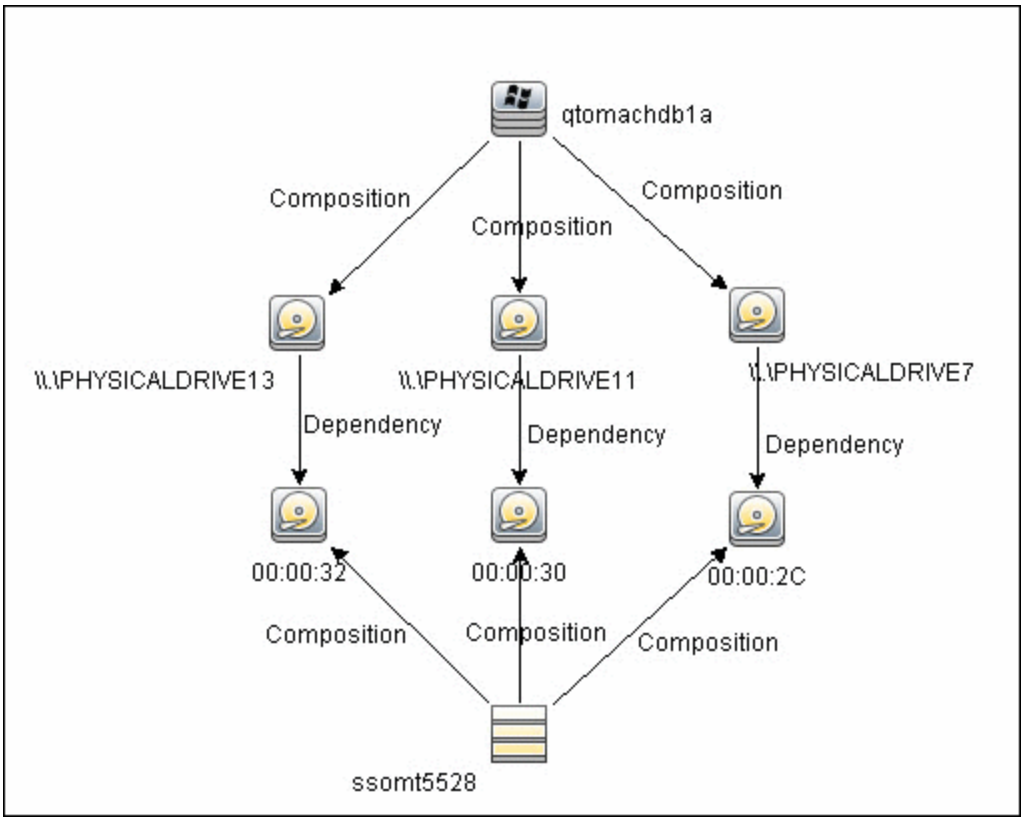
The integration includes the **ECC_Integration.zip** package, which contains the trigger TQL, DFM script, adapter, and job for ECC integration.

Supported Versions

Target Platform	OS Platform	DFM Protocol	ECC Version
EMC Control Center	All	Generic DB (SQL) over JDBC, SSL optional	6.0 and 6.1

Topology

The following diagram illustrates the storage topology and shows the relationships between logical volumes on a storage array and those on servers:



How to Run the ECC/UCMDB Integration Job

This task includes the steps to run the ECC/UCMDB integration job. There are two versions:

- "Run the Job - UCMDB 9.04 and Later" below
- "Run the Job - UCMDB 9.03 and 9.02" on next page

Run the Job - UCMDB 9.04 and Later

Note: For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

In DFM, in the Integration Studio, create a new integration point.

1. Provide a name and description for the integration point.
2. Under **Integration Properties > Adapter**, select the **EMC Control Center** adapter.
3. Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.

4. Under **Adapter Properties > Trigger CI instance** select:
   a. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or
   b. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

   Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

5. Verify the credentials for the chosen CI instance. Right-click on **Trigger CI instance** and select **Actions > Edit Credentials Information**.

   Note: For details about the credentials, see "How to Run the ECC/UCMDB Integration Job" on previous page

6. Save the integration point.
7. Run the job.

Tip: You can include the ECC job in the DFM schedule. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

To connect to the ECC Oracle database with SSL communication, see "How to Run the ECC/UCMDB Integration Job" on previous page.

Run the Job - UCMDB 9.03 and 9.02

1. If you are connecting to the ECC Oracle database with SSL communication, in DFM populate the Generic DB (SQL) protocol parameters with the credentials to the ECC database.
   a. In the Database Type box, choose **oracle**.
   b. Get the **user.crt** certificate file from the Oracle server containing **RAMBDB**.
   c. Build a java trust store file using keytool and the **user.crt** file. Use the following procedures, replacing "----------" throughout with the appropriate information:

```
<hp>\DataFlowProbe\bin\jre\lib\security>..\..\bin\keytool -
genkey -alias eccdb -keyalg RSA -keystore eccstore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
 [Unknown]: ---------------------------------
What is the name of your organizational unit?
 [Unknown]: ---------------------------------
```

```
What is the name of your organization?
 [Unknown]: --------------------------------
What is the name of your City or Locality?
 [Unknown]: --------------------------------
What is the name of your State or Province?
 [Unknown]: --------------------------------
What is the two-letter country code for this unit?
 [Unknown]: --------------------------------
Is ------------------------------------------------------
correct?
 [no]: y

Enter key password for <probe$gt;
       (RETURN if same as keystore password):

<hp>\DataFlowProbe\bin\jre\lib\security>..\..\bin\keytool -
import -keystore eccstore.jks -file user.crt
Enter keystore password:
Owner: CN=chapecc1, OU=ECC Database (Class 2), C=US
Issuer: CN=chapecc, OU=ControlCenter - CLASS2 ROOT, O=EMC, C=US
Serial number: ------------------------------------------------
----------------------------------------------------------------
--------------------------------------------------
Valid from: Tue Dec 29 15:34:17 EST 2009 until: Thu Dec 29
15:34:17 EST 2016
Certificate fingerprints:
       MD5: --------------------------------
       SHA1: -------------------------------
       Signature algorithm name: SHA1withRSA
       Version: 3
Trust this certificate? [no]: y
Certificate was added to keystore

<hp>\DataFlowProbe\bin\jre\lib\security>..\..\bin\keytool -list
-keystore eccstore.jks
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

eccdb, Oct 20, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): --------------------------------
mykey, Oct 20, 2011, trustedCertEntry,
Certificate fingerprint (MD5): --------------------------------
```

d. Specify this trust store file location and password in the Generic DB Protocol (SQL) with

the necessary credentials.

These credentials should have SELECT permissions on the following tables/views:

- Fibre channel switches: **STSSYS.STS_SWITCH_LIST**
- Fibre channel ports on switches: **STSSYS.STS_SWITCH_PORT**
- Storage arrays: **STSSYS.STS_ARRAY_LIST**
- Fibre channel ports on arrays: **STSSYS.STS_ARAY_PORT**
- Logical volumes on arrays: **STSSYS.STS_ARRAY_DEVICE**
- Hosts/servers: **STSSYS.STS_HOST_LIST**
- Fibre channel ports and HBAs on hosts: **STSSYS.STS_HOST_HBA**
- Logical volumes on hosts: **STSSYS.STS_HOST_DEVICE**
- Logical volume dependencies: **STSSYS.STS_HOST_SHAREDDEVICE**
- Port connections: **STSSYS.STS_ARRAY_PORT_CONNECTION**

For credential information, see "Supported Protocols" on page 82.

Note: The ECC database instance has an out-of-the-box user account named **STSVIEW** that includes the necessary privileges. The default password for this account is **sts**.

2. **Prerequisite - Other**

   Verify that the IP address of the ECC server is within scope of a Data Flow Probe. For details, see "Add/Edit IP Range Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

3. **Run the job - UCMDB 9.03 and 9.02**

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

   a. In DFM, in the Discovery Control Panel window, run one of the following sets of jobs to trigger ECC discovery:

   Set 1:

   - **Range IPs by ICMP**. Discovers the IP address of the ECC server.
   - **Host Connection by Shell/WMI/SNMP**. Discovers operating system information on the ECC server.
   - **Host Resources and Applications by Shell/SNMP/WMI**. Discovers the Oracle database instance used by ECC.

- **Oracle Database Connections by SQL**. Discovers Oracle databases using the Generic DB Protocol (SQL).

Set 2:

- **Range IPs by ICMP**. Discovers the IP address of the ECC server.
- **Databases TCP ports**.
- **Oracle Database Connections by SQL**. Discovers Oracle databases using the Generic DB Protocol (SQL).

b. Activate the **Integration – EMC Control Center > ECC Integration by SQL** job. This job discovers the storage infrastructure of ECC.

The **ECC Integration by SQL** job runs SQL queries on the ECC Oracle database using JDBC. This Oracle database instance is used as a trigger for the DFM job. For details, see "ECC Integration Mechanism" below.

Tip: You can include the ECC job in the DFM schedule. For details, see "Discovery Scheduler Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

ECC Integration Job

This section includes:

ECC Integration Mechanism

The following workflow explains how the **ECC Integration by SQL** job discovers the storage topology of ECC. The job:

1. Connects to the ECC Oracle database instance using credentials from the Generic DB Protocol (SQL). For details, see "How to Run the ECC/UCMDB Integration Job" on page 522.
2. Queries for fibre channel switches and ports on each switch and creates **Fibre Channel Switch** CIs:

```
SELECT switch.st_id, switch.st_sn, switch.st_alias, switch.st_
model, switch.st_version, switch.st_vendor, switch.sw_
managementurl, switch.sw_domain, switch.sw_portcount, switch.sw_
portcount_free FROM stssys.sts_switch_list switch WHERE
LOWER(switch.sw_principal) = 'true'
```

3. Queries for fibre channel adapters and ports on each Fibre Channel Switch and creates **Fibre Channel HBA** and **Fibre Channel Port** CIs:

```
SELECT port.port_id, port.port_number, port.port_type, port.adport_
alias, port.port_wwn, port.port_status, port.conn_port_wwn FROM
stssys.sts_switch_port port WHERE port.st_id = switch.st_id from
above query
```

4. Queries for storage arrays and creates **Storage Array** CIs:

```
SELECT array.st_id, array.st_sn, array.st_alias, array.st_type,
array.st_model, array.st_vendor, array.st_microcode, array.sy_
microcode_patch, array.sy_microcode_patchdate FROM stssys.sts_
array_list array
```

5. Queries for Fibre Channel ports, Fibre Channel host bus adapters (HBA), and logical volumes on each storage array, and creates **Fibre Channel Port**, **Fibre Channel Port HBA**, and **Logical Volume** CIs:

```
SELECT port.port_id, port.port_number, port.port_type, port.adport_
alias, port.port_wwn, port.port_status FROM stssys.sts_array_port
port WHERE port.st_id = array.st_id from above query
SELECT hba.port_id, hba.ad_id, hba.ad_name FROM stssys.sts_array_
port hba WHERE hba.st_id = array.st_id from above query
SELECT logicalVolume.sd_id, logicalVolume.sd_name,
logicalVolume.sd_alias, logicalVolume.sd_size, logicalVolume.sd_
type FROM stssys.sts_array_device logicalVolume WHERE
logicalVolume.st_id = array.st_id from above query
```

6. Queries for hosts/servers and creates appropriate **Computer**, **Windows**, or **Unix** CIs. Results of this query are used to create host resource CIs, such as **CPU,** if this information is available:

```
SELECT host.host_id, host.host_name, host.host_alias, host.host_
domain, host.host_model, host.host_ip, host.host_vendorname,
host.host_cpucount, host.host_installedmemory, host.host_os,
host.host_osversion, host.host_oslevel, host.host_osclass FROM
stssys.sts_host_list host
```

7. Queries for Fibre Channel ports, Fibre Channel host bus adapters (HBA), and logical volumes on each host/server and creates **Fibre Channel Port**, **Fibre Channel Port HBA**, and **Logical Volume** CIs:

```
SELECT port.port_id, port.port_number, port.adport_alias,
port.port_wwn FROM stssys.sts_host_hba port WHERE port.host_id =
host.host_id from above query
SELECT hba.ad_id, hba.ad_name, hba.fibread_nodewwn, hba.ad_vendor,
hba.ad_revision, hba.ad_model, hba.port_id, hba.ad_driver_rev FROM
stssys.sts_host_hba hba WHERE hba.host_id = host.host_id from above
query
SELECT logicalVolume.hd_id, logicalVolume.hd_name,
logicalVolume.hd_type, logicalVolume.hd_total FROM stssys.sts_host_
device logicalVolume WHERE logicalVolume.hd_id IS NOT NULL AND
logicalvolume.arrayjbod_type = 'Array' AND logicalVolume.host_id =
host.host_id from above query
```

8. Queries for logical volume mapping between logical volumes on hosts/servers and logical volumes on storage arrays, and adds **Dependency** relationships between hosts/servers and storage arrays:

   ```
   SELECT sd_id FROM stssys.sts_host_shareddevice WHERE hd_id =
   logicalvolume.hd_id from above query
   ```

9. Queries for paths between hosts/servers and storage arrays and adds **Fibre Channel Connect** relationships between respective hosts/servers, switches, and storage arrays:

   ```
   SELECT port.port_wwn, port.conn_port_wwn FROM stssys.sts_array_
   port_connection port WHERE port.port_wwn IS NOT NULL AND port.conn_
   port_wwn IS NOT NULL
   SELECT port.port_wwn, port.conn_port_wwn FROM stssys.sts_switch_
   port port WHERE port.port_wwn IS NOT NULL AND port.conn_port_wwn IS
   NOT NULL
   ```

Trigger Query

Trigger CI: ECC Oracle database

Adapter

- **Adapter Parameters**

Parameter	Description
allowDNSLookup	If a node in the ECC database does not have an IP address but has a DNS name, it is possible to resolve the IP address by the DNS name. ▪ **True:** If a node does not have an IP address, an attempt is made to resolve the IP address by DNS name (if a DNS name is available). **Default:** False
ignoreNodesWithoutIP	Defines whether or not nodes in ECC without IP addresses should be pulled into UCMDB. ▪ **True**. Nodes without IPs are ignored. ▪ **False.** A Node CI is created with an ECC ID as the node key attribute. **Note:** Setting this parameter to **False** may result in duplicate CIs in the CMDB. **Default:** True

Discovered CITs and Relationships

- **CPU**
- **Containment**

- **Composition (link)**
- **Dependency (link)**
- **Fibre Channel Connect (link)**
- **Fibre Channel HBA**
- **Fibre Channel Port**
- **Fibre Channel Switch**
- **Node**
- **IpAddress**
- **Logical Volume**
- **Membership (link)**
- **Storage Array**
- **Storage Processor**
- **Unix**
- **Windows**

Views

The **Storage_Basic** package contains views that display common storage topologies. These are basic views that can be customized to suit the integrated ECC applications.

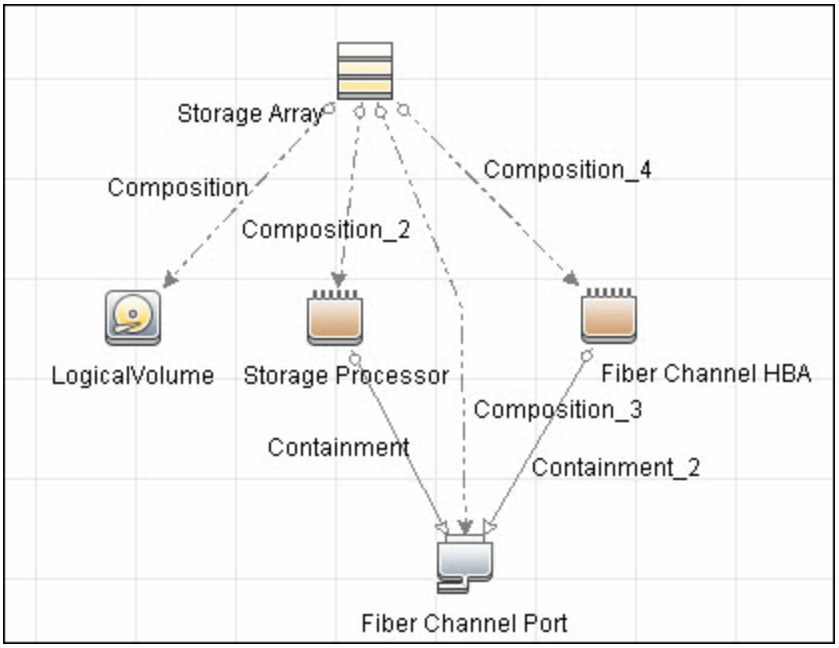
To access the Storage_Basic package: **Administration > Package Manager**. For details, see "Package Manager" in the *HP Universal CMDB Administration Guide*.

This section includes:

Storage Array Details

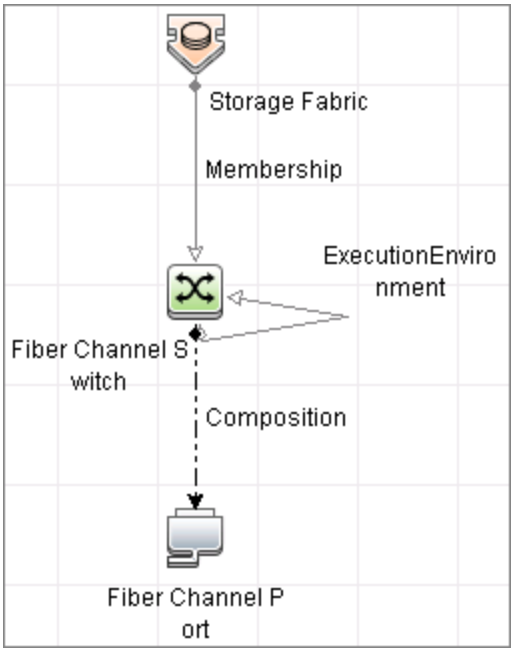
This view shows a Storage Array and its components including Logical Volumes, HBAs, Storage Processors, and Fibre Channel Ports. The view shows each component under its container Storage Array and groups Logical Volumes by CI Type.

Storage Array does not require all components in this view to be functional. Composition links stemming from the Storage Array have a cardinality of zero-to-many. The view may show Storage Arrays even when there are no Logical Volumes or Storage Processors.



FC Switch Details

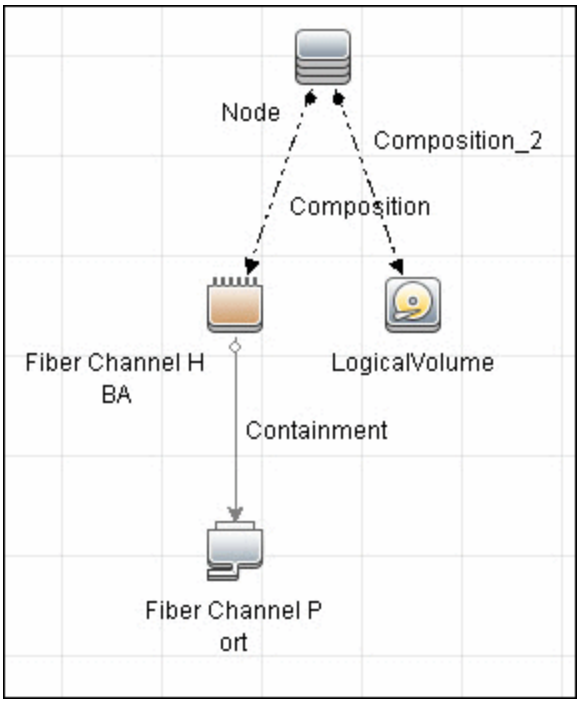
This view shows a Fibre Channel Switch and all connected Fibre Channel Ports.



Note: Although shown in the preceding graphic, the ECC job does not discover Storage Fabrics. The view represented by this query is populated without Storage Fabrics.

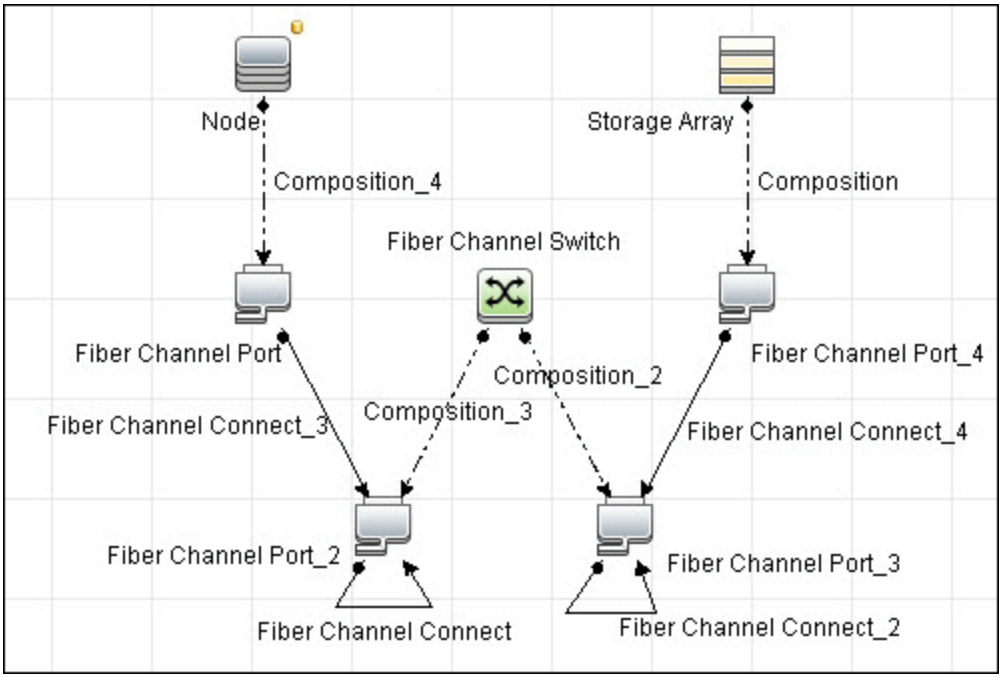
Host Storage Details

This view shows only Hosts that contain a Fibre Channel HBA or a Logical Volume. This keeps the view storage-specific and prevents hosts discovered by other DFM jobs from being included in the view.



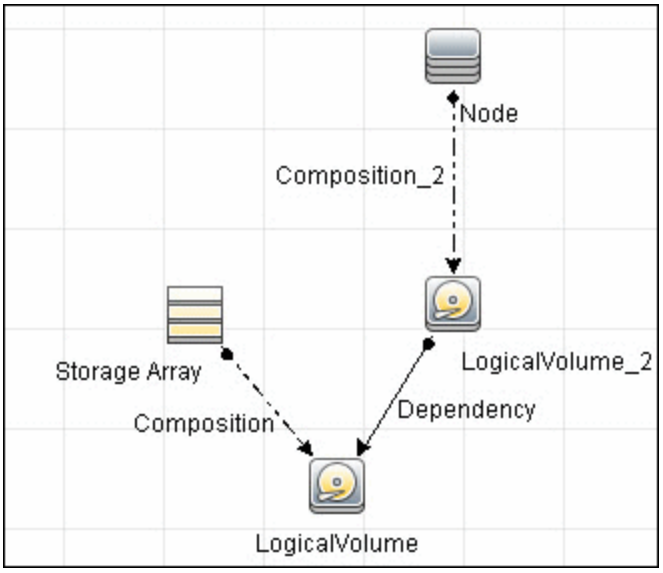
SAN Topology

This view maps physical connections between Storage Arrays, Fibre Channel Switches, and Hosts. The view shows Fibre Channel Ports below their containers. The view groups the Fibre Channel Connect relationship CIT to prevent multiple relationships between the same nodes from appearing in the top layer.



Storage Topology

This view maps logical dependencies between Logical Volumes on Hosts and Logical Volumes on Storage Arrays. There is no folding in this view.



Impact Analysis Rules

The **Storage_Basic** package contains basic impact analysis rules to enable impact analysis and root cause analysis in UCMDB. These impact analysis rules are templates for more complex rules that you can define based on business needs.

All impact analysis rules fully propagate both Change and Operation events. For details on impact analysis, see "Impact Analysis Manager Page" and "Impact Analysis Manager Overview" in the *HP Universal CMDB Modeling Guide*.

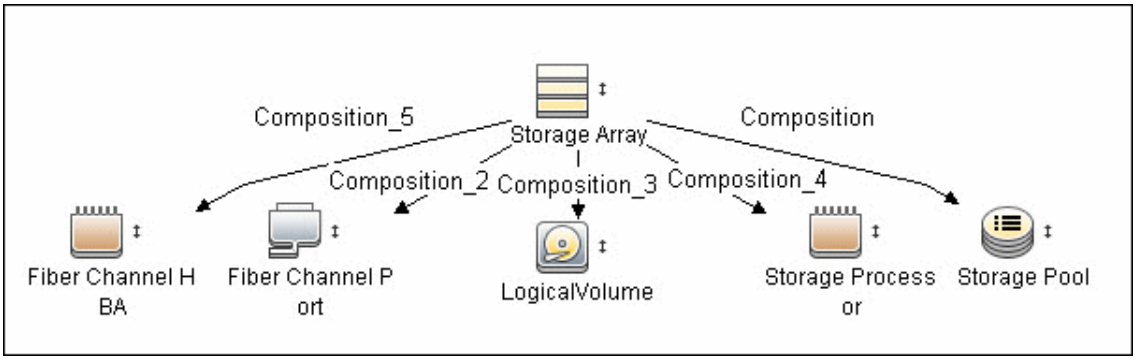
To access the Storage_Basic package: **Administration > Package Manager**. For details, see "Package Manager" in the *HP Universal CMDB Administration Guide*.

Note: Impact analysis events are not propagated to Fibre Channel Ports for performance reasons.

This section includes:

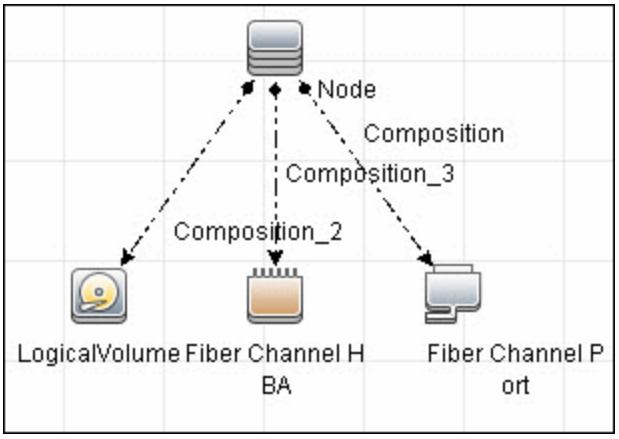
Storage Array Devices to Storage Array

This impact analysis rule propagates events between Logical Volumes, Storage Processors, Fibre Channel HBAs, and Storage Arrays.



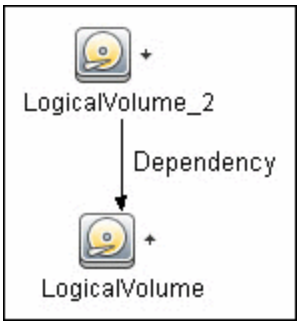
Host Devices to Host

This impact analysis rule propagates events between Fibre Channel HBAs and Hosts, and Logical Volumes on the Host.



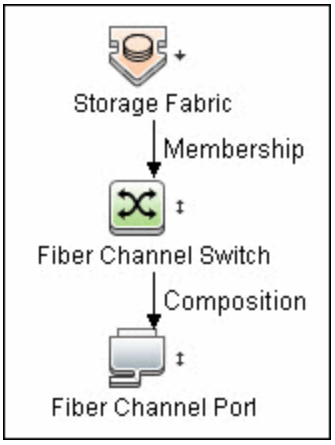
Logical Volume to Logical Volume

This impact analysis rule propagates events on a Logical Volume contained in a Storage Array to the dependent Logical Volume on the Host.



FC Switch Devices to FC Switch

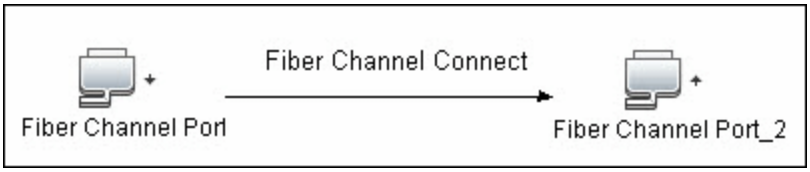
This impact analysis rule propagates events from a Fibre Channel Port to and from a Switch. The event is also propagated to the associated Storage Fabric.



Note: Although shown in the preceding graphic, the ECC job does not discover Storage Fabrics. The rule represented by this query is used without Storage Fabrics.

FC Port to FC Port

This rule propagates events on a Fibre Channel Port to another connected Channel Port.



Example Scenario of HBA Crashing on a Storage Array

- The event propagates from the HBA to the Storage Array and the Logical Volumes on the Array because of the Storage Devices to Storage Array rule.
- The impact analysis event on the Logical Volume then propagates to other dependent Logical Volumes through the Logical Volume to Logical Volume rule.
- Hosts using those dependent Logical volumes see the event next because of the Host Devices to Host rule.
- Depending on business needs, you define impact analysis rules to propagate events from these hosts to applications, business services, lines of business, and so on. This enables end-to-end mapping and impact analysis using UCMDB.

Reports

The **Storage_Basic** package contains basic reports that can be customized to suit the integrated ECC applications.

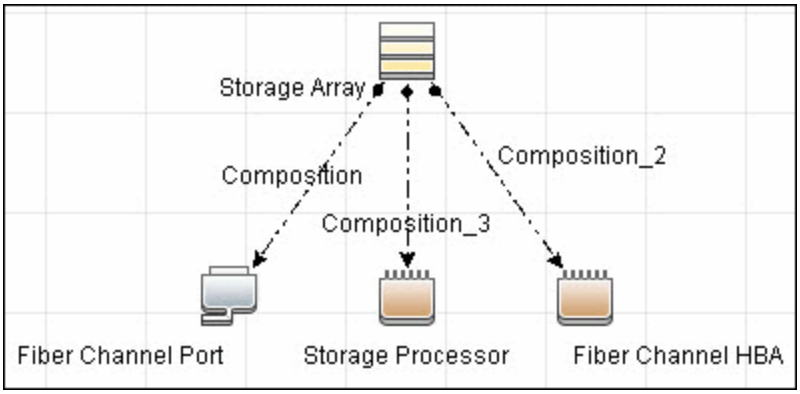
In addition to the system reports, Change Monitoring and Asset Data parameters are set on each CIT in this package, to enable Change and Asset Reports in UCMDB.

To access the Storage_Basic package: **Administration > Package Manager**. For details, see "Package Manager" in the *HP Universal CMDB Administration Guide*.

This section includes:

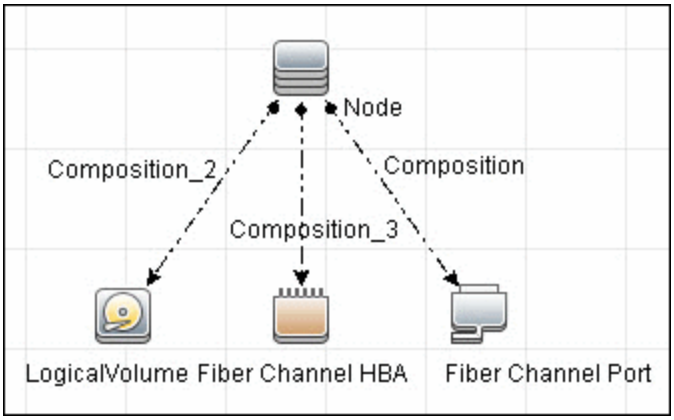
Storage Array Configuration

This report shows detailed information on Storage Arrays and its sub-components including Fibre Channel Ports, Fibre Channel Arrays, and Storage Processors. The report lists Storage Arrays with sub-components as children of the Array.



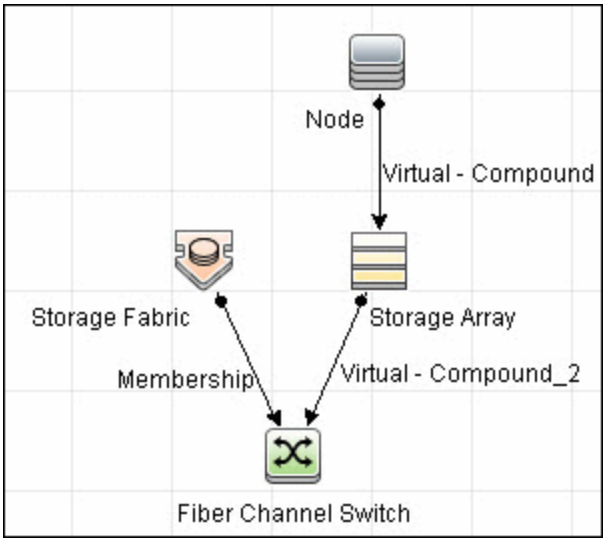
Host Configuration

This report shows detailed information on hosts that contain one or more Fibre Channel HBAs, Fibre Channel Ports, or Logical volumes. The report lists hosts with sub-components as children of the host.



Storage Array Dependency

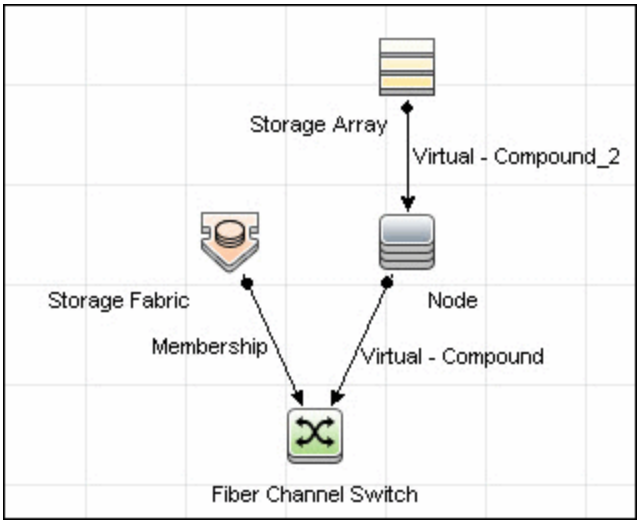
This report maps dependencies on a Storage Array. The report also displays information on switches connected to it.



Note: Although shown in the preceding graphic, the ECC job does not discover Storage Fabrics. The report represented by this query is populated without Storage Fabrics.

Host Storage Dependency

This report shows detailed information on storage infrastructure dependencies of a Host. The report lists hosts and dependent components.



Note: Although shown in the preceding graphic, the ECC job does not discover Storage Fabrics. The report represented by this query is populated without Storage Fabrics.

Chapter 44

Federating KPI Data from Configuration Manager

This chapter includes:

Overview

The federation mechanism that is built into HP Universal CMDB enables UCMDB to be used as a contact repository for sharing data among external applications, without duplicating it. By federating data from Configuration Manager to UCMDB, external applications can consume its analysis information in various ways:

- Use UCMDB's reporting functionality to generate and schedule reports on top of Configuration Manager's data.
- Consume Configuration Manager's data in other HP applications, such as HP Business Service Management.
- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

Configuration Manager exposes the following data for federation:

- **Policy compliance status** data includes information about current policy result data for managed CIs and the associated policies.
- **Authorization status** data includes information about the authorization status of managed CIs.

UCMDB provides the class model for the schema for the model to be shared, and uses a federation TQL query as the way to consume data in UCMDB on the fly. For details, see "Federating KPIs" in the *HP Universal CMDB Configuration Manager User Guide*.

KPI means **Key Performance Indicator**.UCMDB provides the **CMKpiAdapter** to federate KPI information about policy and authorization status from Configuration Manager. The data that is federated from Configuration Manager populates the **Kpi** and **KpiObjective** CITs, and can be retrieved by TQL as described in "Create KPI Reports" on page 542.

How to Consume Federated KPI Data from Configuration Manager

This workflow provides a brief overview of the steps to be performed in UCMDB, in order to consume federated Kpi data from Configuration Manager.

This task includes the following steps:

- "Create an Integration Point to Federate KPI Data" below
- "Create KPI Reports" on page 542

Create an Integration Point to Federate KPI Data

3. In DFM, in the Integration Studio, create a new integration point.
4. Set the following adapter properties:

Field	Description
Adapter	Click  and select **CMKpiAdapter**.
Credentials ID	Do the following: a. Click  . b. Select **Generic Protocol** and click **OK**. c. Click  to add the credentials to connect to the Configuration Manager database. Enter credentials for the user who has Views Administration and Login permissions. d. When finished, click **OK**
Hostname/IP	Provide the host name or IP address of the Configuration Manager database
Integration Name	Enter a name for the new integration point.
Port	Enter the port number that is used for communication with the Configuration Manager database.
Use SSL	Select **False**. You cannot use secured communication to federate data from Configuration Manager.

5. Click **Test Connection** to make sure that you have configured the integration point correctly.
6. Click **OK** to save the integration point.
7. Select the KPI and KPIObjective CI types in the Supported and Selected CI Types tree.
8. Click  to save the integration point.

For further details about creating integration points, see the section about the Integration Studio in the *HP Universal CMDB Data Flow Management Guide*.

Create KPI Reports

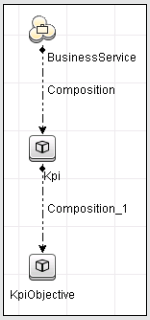
You can create KPI reports based on the CIs in a view, a custom TQL query, or business services.

1. In UCMDB, create a new view based on a custom TQL or copy an existing view.

   Note: When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see "Troubleshooting and Limitations" (on page 1).

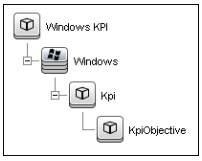
2. For each configuration item that you want to associate with a policy, attach the selected CI to the Kpi CI type and the Kpi CI type to the KpiObjective CI type, using composition links. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated KPI information.

   Note: If you want to create a business services report, select the BusinessService CI type when creating the TQL query.

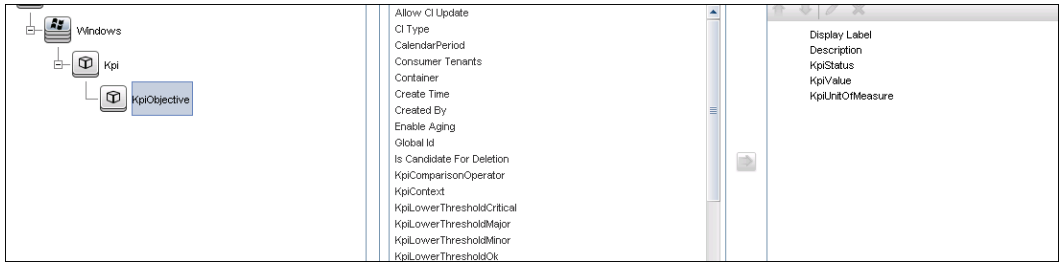
   

3. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.

4. Set the hierarchy. An example is shown below:

5. Add properties for the KpiObjective CI type to the report layout: An example is shown below:

6. If desired, you can schedule these reports to be created periodically. For details, see the *HP Universal CMDB Data Flow Management Guide*.

For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

Troubleshooting and Limitations

- Federation only works with CIs in the actual state. Therefore:
  - Policy compliance is federated only for CIs in the actual state.
  - The authorization status for CIs that were deleted from the actual state is not shown.
- SSL communication for KPI data federation is not supported.
- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the Max Num To Federate setting in the Infrastructure Settings Manager in UCMDB. For details about changing settings, see the Infrastructure Settings Manager chapter in the *HP Universal CMDB Administration Guide*. The recommended number of CIs is no more than 20,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the *HP Universal CMDB Configuration Manager Deployment Guide*.
- If the test connection fails, click **Details** and check the first error in the stack trace for more information.

Chapter 45

Federating Policy Data from Configuration Manager

This chapter includes:

Overview

The federation mechanism that is built into HP Universal CMDB enables UCMDB to be used as a contact repository for sharing data among external applications, without duplicating it. By federating data from Configuration Manager to UCMDB, external applications can consume its analysis information in various ways:

- Use UCMDB's reporting functionality to generate and schedule reports on top of Configuration Manager's data.
- Consume Configuration Manager's data in other HP applications, such as HP Business Service Management.
- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

Configuration Manager exposes **Policy compliance status** data (which includes information about current policy result data for managed CIs and the associated policies) for federation.

UCMDB provides the class model for the schema for the model to be shared, and uses a federation TQL query as the way to consume data in UCMDB on the fly. For details, see "Federating Policy Compliance Data" in the *HP Universal CMDB Configuration Manager User Guide*.

UCMDB provides the **CMPolicyAdapter** to federate policy data from Configuration Manager, which populates the **Policy** and **PolicyResult** CITs, and can be retrieved by TQL as described in "Create Policy Reports Based on CIs in a View or Custom TQL query" on next page and "Create summary policy reports based on the CIs in a view or a custom TQL query" on page 548.

How to Consume Federated Policy Data from Configuration Manager

This workflow provides a brief overview of the steps to be performed in UCMDB, in order to consume federated data from Configuration Manager.

This task includes the following steps:

- "Create an Integration Point to Federate Policy Compliance Data" below
- "Create Policy Reports Based on CIs in a View or Custom TQL query" on next page
- "Create summary policy reports based on the CIs in a view or a custom TQL query" on page 548

Create an Integration Point to Federate Policy Compliance Data

4. In DFM, in the Integration Studio, create a new integration point.
5. Set the following adapter properties:

Field	Description
Adapter	Click  and select **CMPolicyAdapter**.
Credentials ID	Do the following: a. Click  . b. Select **Generic DB Protocol (SQL)** and click **OK**. c. Click  to add the credentials to connect to the Configuration Manager database. These should be the same credentials that were provided during the installation of Configuration Manager. d. When finished, click **OK**
DB Name/SID	The database name or schema ID.
DB Type	Specify Oracle or MSSQL, as required.
Hostname/IP	Provide the host name or IP address of the Configuration Manager database
Integration Name	Enter a name for the new integration point.
Port	Enter the port number that is used for communication with the Configuration Manager database.

6. Click **Test Connection** to make sure that you have configured the integration point correctly. If the test fails, see "Troubleshooting and Limitations" (on page 1).
7. Click **OK** to save the integration point.
8. Select the Policy and PolicyResults CI types in the Supported and Selected CI Types tree.
9. Click  to save the integration point.

For further details about creating integration points, see the section about the Integration Studio in the *HP Universal CMDB Data Flow Management Guide*.

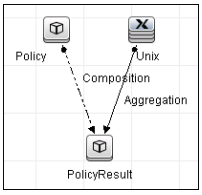
Create Policy Reports Based on CIs in a View or Custom TQL query

1. Create an integration point as described in "Create an integration point to federate policy compliance data" (on page 1), if one does not already exist.
2. In UCMDB, create a new view with a custom TQL query, or copy an existing view.

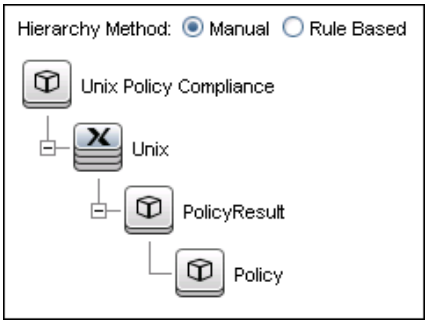
   Note: When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see "Troubleshooting and Limitations" (on page 1).

3. For each configuration item that you want to associate with a policy, attach the Policy CI type

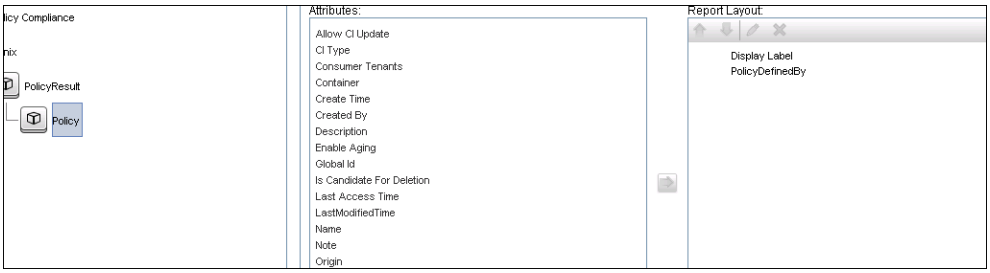
and the selected CI to the PolicyResult CI type, using composition and aggregation links accordingly. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated policy information. An example is shown below:



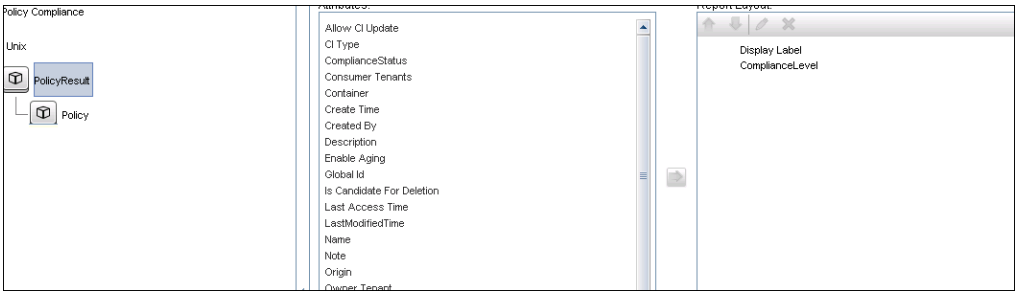
4. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.
5. Set the hierarchy. An example is shown below:



6. Add properties for the Policy CI type to the report layout: An example is shown below:



7. Add properties for the PolicyResult CI type to the report layout. An example is shown below:



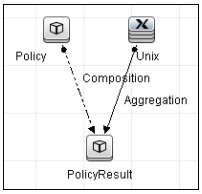
8. If desired, you can schedule these reports to be created periodically. For details, see the *HP Universal CMDB Data Flow Management Guide*.

For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

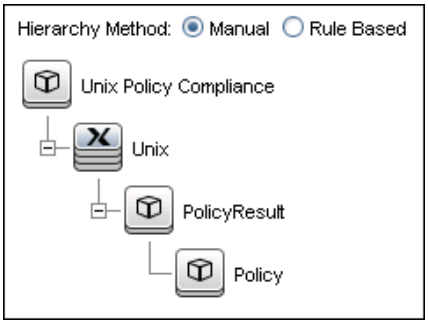
Create summary policy reports based on the CIs in a view or a custom TQL query

1. Create an integration point as described in "Create an integration point to federate policy compliance data" (on page 1), if one does not already exist.
2. In UCMDB, create a new view or copy an existing view.

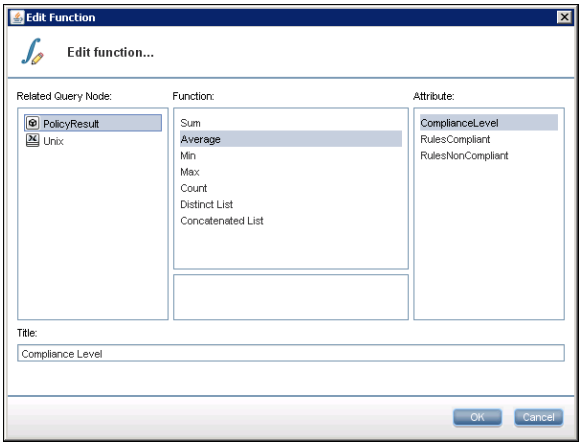
   Note: When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account. For details, see "Troubleshooting and Limitations" (on page 1).
3. For each configuration item that you want to associate with a policy, attach the Policy CI type and the selected CI to the PolicyResult CI type, using composition and aggregation links accordingly. The cardinality should be 0..* if you also want to obtain results for CIs that do not have associated policy information. An example is shown below:



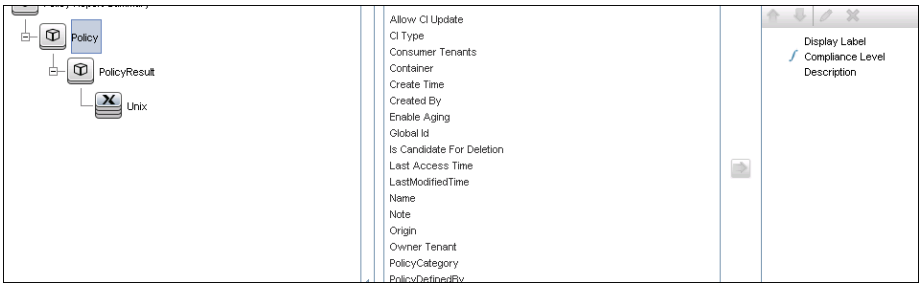
4. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.
5. Set the hierarchy. An example is shown below:



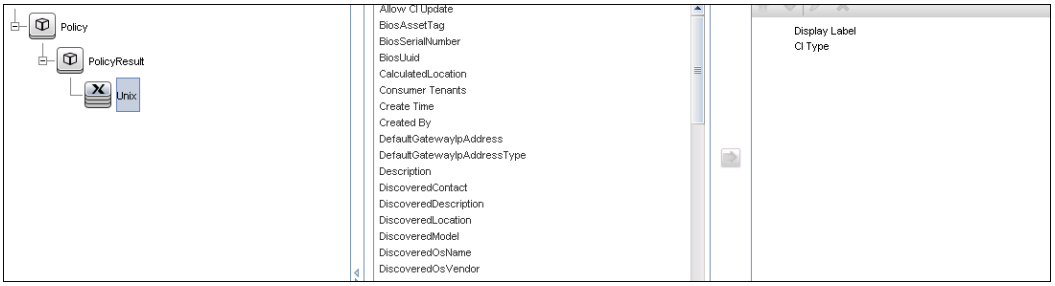
6. Create an aggregation function for the Policy CI type. An example is shown below:



7. Add properties for the Policy CI type to the report layout. An example is shown below:



8. Add properties for the ConfigurationItem CI type to the report layout. An example is shown below:



9. Change the report format to a bar chart. An example is shown below:

10. If desired, you can schedule these reports to be created periodically. For details, see *HP Universal CMDB Data Flow Management Guide*.

For details about creating reports, see the section about reports in the *HP Universal CMDB Modeling Guide*.

Troubleshooting and Limitations

- Federation only works with CIs in the actual state. Therefore:
  - Policy compliance is federated only for CIs in the actual state.
  - The authorization status for CIs that were deleted from the actual state is not shown.
- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the Max Num To Federate setting in the Infrastructure Settings Manager in UCMDB. For details about changing settings, see the Infrastructure Settings Manager chapter in the *HP Universal CMDB Administration Guide*. The recommended number of CIs is no more than 20,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the *HP Universal CMDB Configuration Manager Deployment Guide*.
- If the test connection fails, click **Details** and check the first error in the stack trace for more information.

Chapter 46

HP ServiceCenter/Service Manager Integration

This chapter includes:

Note: This adapter is a specific configuration of the ServiceDesk Adapter.

Overview

The ServiceCenter/Service Manager adapters support the push to and retrieval of data from HP ServiceCenter and HP Service Manager. These adapters connect to, send data to, and receive data from ServiceCenter/Service Manager using the Web Service API. Every request to ServiceCenter/Service Manager to calculate a federated query or to push data is made through these adapters. These adapters are compatible with HP ServiceCenter version 6.2, and HP Service Manager, versions 7.0x, 7.1x, and 7.2x-9.2x (following changes to the WSDL configuration).

The adapters are provided with preconfigured jobs to transfer Incident, Problem, and Planned Change CI types between ServiceCenter/Service Manager and UCMDB.

Data Push

Note: The Data Push flow is relevant for HP Service Manager version 7.1 and later only.

The data push framework uses the adapter to push CIs and relationships to HP Service Manager. Once a CI has been pushed to HP Service Manager, an Actual State flow may be triggered in HP Service Manager, and selecting a tab in HP Service Manager enables you to view the most updated data available on the CI in UCMDB.

For details about setting up a data push flow, see "Data Push Tab" in the *HP Universal CMDB Data Flow Management Guide*.

Federation

The adapter supports three external CI types: Incident, Problem, and Planned Change. The adapter retrieves the CIs of these types from ServiceCenter/Service Manager with the required layout and by a given filter (using reconciliation and/or a CI filter). Each of these CITs can be related to one of the following UCMDB internal CITs: Host, Business Service, Application. Each UCMDB internal CIT includes a reconciliation rule in the ServiceCenter/Service Manager configuration that can be changed dynamically. For details, see "Reconciliation Data Configuration" on page 562. Note that there are no internal relationships between adapter-supported CITs.

The modeling of the supported CITs and virtual relationships is supplied with the Adapter. You can add attributes to a CIT. For details, see "How to Add an Attribute to the ServiceCenter/Service Manager CIT" on page 573.

For details about setting up a federation flow, see "Federation Tab" in the *HP Universal CMDB Data Flow Management Guide*.

Supported Versions

UCMDB is delivered with four different Service Manager adapters, for different versions of HP ServiceCenter/HP Service Manager. When you define an integration, choose the correct adapter according to your Service Manager version.

For documentation about **ServiceManagerAdapter9.x**, click the help button after selecting that adapter (**Integration Studio > Integration Point > New Integration Point > Adapter**).

Data Push Flow

You can configure the data push flow options for the Service Manager integration by updating the following UCMDB, Service Manager and adapter XML files:

- **xslt files.** Maps the UCMDB graph to the Service Manager request.
- **smSyncConfFile.** Maps a tql name to an xslt file. This resource should be changed when adding a new TQL query.

Multi-Threading

By default, the ServiceDesk Adapter uses six concurrent threads to push data to Service Manager. To configure the ServiceDesk Adapter multi-thread settings, edit the **sm.properties** file, located in:

Data Flow Management > Adapter Management > ServiceManagerAdapter corresponding to Service Manager version > Configuration Files

Error Handling

The ServiceCenter/Service Manager adapter has a mechanism that permits the capture of CIs that failed in a push job due to specific errors, and instead of failing the entire push job, attempts to send them again in future executions. In such a case, the statistics display the **Successful with warnings** status.

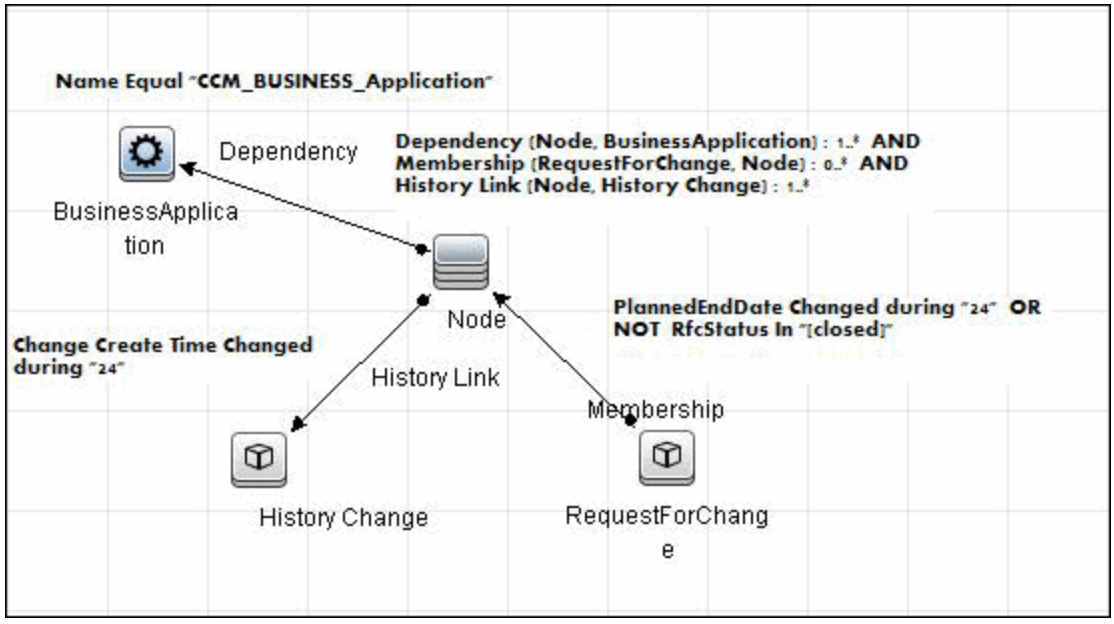
By default, only the error of `locked CI` (Error 3) triggers this mechanism.

To configure error handling, navigate to **Adapter Management > ServiceManagerAdapterX-X > Configuration Files > sm.properties** and set the required values.

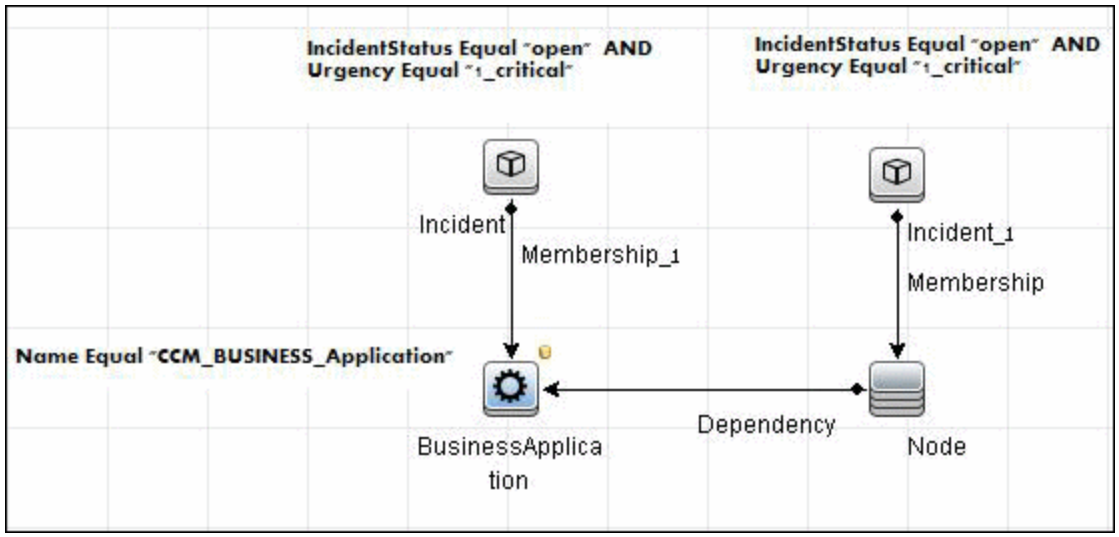
Federation Use Cases

The following use cases (which include TQL query examples) describe how the adapter can be used:

- A user needs to display all unplanned changes to all hosts running a specific application during the last 24 hours:



- A user needs to see all open critical incidents on an application and its hosts:



Viewing the Actual State

UCMDB exposes a Web Service for the use of Service Manager. The Web Service receives the CMDB ID and customer ID as input and returns extended data for the CI, which includes properties and related CIs.

The call to the Web Service is done in the Actual State tab in HP Service Manager, when Service Manager is configured to work with UCMDB.

The Web Service executes the query in the **Integration\SM Query** folder that matches the type of CI sent. If more than one matching query exists, an exception is thrown.

The layout that is defined in the TQL query is the layout that is synchronized.

It is common for some parts of the executed query to be federated (for example, from DDMi, Asset Manager, SMS, and so on).

This section also includes:

- "Predefined Queries" below
- "Configuration" on next page

Predefined Queries

Out-of-the-box queries are located in the **Integration\SM Query** folder. Queries are selected according to the class type of the CI.

- **hostExtendedData.** Used for retrieving real time extended information (Asset, Person, WindowsService, Printer, InstalledSoftware, and CPU) about a certain CI of type Node.
- **applicationExtendedData.** Used for retrieving real time extended information about Business Applications.
- **businessServiceExtendedData.** Used for retrieving real time extended information about Business Services.

Configuration

WSDL and XML Schema URLs for the Web Service

- **WSDL:**

```
http://[machine_name]:8080/axis2/
services/ucmdbSMService
```

- **XML Schema:**

```
http://[machine_name]:8080/axis2/
services/ucmdbSMService?xsd=xsd0
```

Manipulating the Result Using Transformations

In some cases you may want to apply additional transformations to the resulting XML (for example, to sum up all the disks' sizes and add those as an additional attribute to the CI). To add invoke additional transformation on the TQL results, place a resource named **[tql_name].xslt** in the adapter configuration as follows: <SF> generic?**Adapter Management > ServiceDeskAdapter7-1 > Configuration Files > [tql_name].xslt**.

There is a resource named **example_calculated_attribute.xslt** that demonstrates how to sum the disk sizes using xslt.

Using Global IDs

It is possible to use the Global ID instead of the CMDB ID to work with the Actual State flow. This may be needed in multiple CMDB environments, where a non-CMS UCMDB is integrated with Service Manager. To use global IDs instead of CMDB IDs, navigate to **Adapter Management > ServiceManagerAdapterX-X > Configuration Files > sm.properties** and set **use.global.id=true**.

For details about multiple CMDB environments, see "Integrating Multiple CMDBs" in the *HP Universal CMDB Data Flow Management Guide*.

If CIs were previously pushed to Service Manager from a different CMDB instance, duplicates may occur, as the CIs will not reconcile.

Compressing Location Topology to an Attribute

Due to the limitation of the Data Push flow, it is not possible to push topologies that have CIs that are not connected directly to the Root. To be able to push locations to Service Manager, an enrichment is used to concatenate the location topology to a single attribute (Calculated Location) on the Node.

The enrichments are found in the **Location** folder:

- Location_1Enrhicment
- Location_2Enrichment
- Location_3Enrichment

The xslt transformer then inflates the attribute back to separate XML tags with the following xslt code:

```
<xsl:variable name="calculatedLocation" select="@calculated_
location"/>
    <Building>
        <xsl:value-of select="substring-after($calculatedLocation,'
Building:')"/>
    </Building>
    <Floor>
        <xsl:value-of select="substring-before(substring-
after($calculatedLocation,'Floor:'),' Building:')"/>
    </Floor>
    <Room>
        <xsl:value-of select="substring-before(substring-
after($calculatedLocation,'Room:'),' Floor:')"/>
    </Room>
```

The serviceDeskConfiguration.xml File

The **serviceDeskConfiguration.xml** Adapter configuration file contains three parts:

The first part, which is defined by the **`ucmdbClassConfigurations`** element, contains the external CIT configuration that the Adapter supports. For details, see "External CITs Configuration" on next page.

The second part, defined by the **`reconciliationClassConfigurations`** element, contains reconciliation data information for appropriate UCMDB CITs. For details, see "Reconciliation Data Configuration" on page 562.

The third part, defined by the **`globalConnectorConfig`** element, includes the global configuration for a specific connector implementation. For details, see "Global Configuration" on page 567.

This section also includes the following topics:

External CITs Configuration

Each CIT that is supported by the adapter is defined in the first section of the adapter configuration file.

This section, **ucmdbClassConfiguration**, represents the only supported CIT configuration. This element contains the CIT name as defined in the UCMDB class model (the **ucmdbClassName** attribute), mapping for all its attributes (the **attributeMappings** element), and a private configuration for a specific connector implementation (the **classConnectorConfiguration** element):

- The **ucmdbClassName** attribute defines the UCMDB class model name.
- The **attributeMappings** element contains `attributeMapping` elements.

  The **attributeMapping** element defines the mapping between the UCMDB model attribute name (the **ucmdbAttributeName** attribute) to an appropriate ServiceCenter/Service Manager attribute name (the **serviceDeskAttributeName** attribute).

  For example:

  ```
  <attributeMapping ucmdbAttributeName="problem_brief_description"
  serviceDeskAttributeName="brief.description"/>
  ```

  This element can optionally contain the following converter attributes:

  - The **converterClassName** attribute. This is the converter class name that converts the UCMDB attribute value to the ServiceDesk attribute value.
  - The **reversedConverterClassName** attribute. This is the converter class name that converts the ServiceDesk attribute value to the UCMDB attribute value.
- The **classConnectorConfiguration** element contains the configuration for the specific connector implementation for the current external CIT. Wrap this configuration in CDATA if it contains special XML characters (for example, `&` replacing `&`).

  The useful fields of the Service Manager **classConnectorConfiguration** element are as follows:

  - The **device_key_property_names** element contains the fields names in the WSDL information of the current object that can contain the device ID (for example, **ConfigurationItem**). Each field should be added as a **device_key_property_name** element.
  - The **id_property_name** element contains the field name in the WSDL information that contains the ID of the current object.

The following example shows the **ucmdbClassConfiguration** section of the **serviceDeskConfiguration.xml** file. The section includes the **ucmdbClassName** element for the Incident CIT with a ServiceCenter connector implementation:

```
<ucmdbClassConfiguration ucmdbClassName="it_incident">
            <attributeMappings>
                <attributeMapping ucmdbAttributeName="incident_id"
serviceDeskAttributeName="IncidentID"/>
```

```
                <attributeMapping ucmdbAttributeName="incident_brief_
description" serviceDeskAttributeName="BriefDescription"/>
                <attributeMapping ucmdbAttributeName="incident_
category" serviceDeskAttributeName="Category"/>
                <attributeMapping ucmdbAttributeName="incident_
severity" serviceDeskAttributeName="severity"/>
                <attributeMapping ucmdbAttributeName="incident_open_
time" serviceDeskAttributeName="OpenTime"/>
                <attributeMapping ucmdbAttributeName="incident_update_
time" serviceDeskAttributeName="UpdatedTime"/>
                <attributeMapping ucmdbAttributeName="incident_close_
time" serviceDeskAttributeName="ClosedTime"/>
                <attributeMapping ucmdbAttributeName="incident_status"
serviceDeskAttributeName="IMTicketStatus"/>
            </attributeMappings>
 <classConnectorConfiguration>
                <![CDATA[ <class_configuration connector_class_
name="com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter
.serviceCenterConnector.impl.SimpleServiceCenterObjectConnector">
          <device_key_property_names>
            <device_key_property_name>ConfigurationItem</device_key_
property_name>
          </device_key_property_names>
          <id_property_name>IncidentID</id_property_name>
          <keys_action_info>
            <request_
name>RetrieveUcmdbIncidentKeysListRequest</request_name>
<response_name>RetrieveUcmdbIncidentKeysListResponse</response_name>
          </keys_action_info>
          <properties_action_info>
            <request_name>RetrieveUcmdbIncidentListRequest</request_
name>
            <response_
name>RetrieveUcmdbIncidentListResponse</response_name>
          </properties_action_info>
        </class_configuration> ]]>
            </classConnectorConfiguration>
        </ucmdbClassConfiguration>
```

Adding Attributes to a CIT

To add an attribute to the UCMDB model for an adapter-supported CIT:

1. Navigate to **Data Flow Management > Adapter Management >** and select the **ServiceManagerAdapter** that corresponds to your version of Service Manager.
2. Select **Configuration Files > ServiceDeskConfiguration.xml** file and add an `attributeMapping` element to the appropriate `ucmdbClassConfiguration` element.
3. Verify that ServiceCenter/Service Manager externalizes this attribute in its Web Service API.
4. Click **Save**.

Reconciliation Data Configuration

Each UCMDB CIT that can be related to the adapter-supported CIT is defined in the second section of the **serviceDeskConfiguration.xml** file.

This section, **`reconciliationClassConfigurations`**, represents the reconciliation data configuration for one UCMDB CIT. The element includes the following attributes:

- **ucmdbClassName.** This is the CIT name as defined in the UCMDB class model.
- **concreteMappingImplementationClass.** This is the class name of the concrete implementation for the **`ConcreteMappingEngine`** interface. Use this attribute to map between instances of UCMDB CITs and external Adapter CITs. The default implementation that is used is:

  ```
  com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.mapping.impl.
  OneNodeMappingEngine
  ```

  An additional implementation exists that is used only for the host reconciliation CIT for reconciliation by the IP of the host:

  ```
  com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter
  .mapping.impl. HostIpMappingEngine
  ```

The **`reconciliationClassConfiguration`** element can contain one of the following elements:

- The **`reconciliationById`** element. This element is used when the reconciliation is done by ID. In this case, the text value of this element is the ServiceDesk field name that contains the CMDB ID. For example:

  ```
  <reconciliationById>UcmdbID</reconciliationById>
  ```

  In this example, the ServiceDesk field `UcmdbID` contains the CMDB ID of the appropriate host.

- The **`reconciliationData`** element. This element is used if the reconciliation is done by comparing attributes. You can run reconciliation with one attribute or several attributes by using the logical operators OR and/or AND.

  If you run reconciliation with one attribute, the **`reconciliationData`** child element should be a **`reconciliationAttribute`** element. The **`reconciliationAttribute`** element contains an appropriate UCMDB attribute name (the **`ucmdbAttributeName`** attribute) and an appropriate ServiceDesk attribute name (the **`serviceDeskAttributeName`** attribute). This element can also contain a **`ucmdbClassName`** attribute that defines the appropriate UCMDB CIT name. By default, the current reconciliation UCMDB CIT name is used.

  You can also use the **`converterClassName`** and **`reversedConverterClassName`** attributes; they should contain the converter class name that converts the UCMDB attribute value to the ServiceDesk attribute value, or vice versa.

  For example:

  ```
  <reconciliationData>
   <reconciliationAttribute ucmdbAttributeName="name"
  serviceDeskAttributeName="NetworkName"
  ```

```
converterClassName="com.mercury.topaz.fcmdb.adapters.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
 </reconciliationData>
```

For reconciliation to run with two or more attributes, use a logical operator between reconciliation attributes.

The logical operator AND can contain several **reconciliationAttribute** elements (the minimum is 2). In this case the reconciliation rule contains an AND operator between attribute comparisons.

For example:

```
<reconciliationData>
<AND>
 <reconciliationAttribute ucmdbAttributeName="name"
serviceDeskAttributeName="NetworkName"
converterClassName="com.mercury.topaz.fcmdb.adapters.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
<reconciliationAttribute ucmdbClassName="ip_address"
ucmdbAttributeName="name" serviceDeskAttributeName="NetworkAddress"
/>
</AND>
</reconciliationData>
```

In this example, the reconciliation rule follows this format: **node.name= NetworkName** and **ip_address.name= NetworkAddress**.

The logical operator OR can contain several **reconciliationAttribute** and AND elements. In this case, the reconciliation rule contains an OR operator between attributes and AND expressions. Since XML does not assure the order of elements, you should provide a priority attribute to each sub-element of OR element type. The comparison between OR expressions is calculated by these priorities.

For example:

```
<reconciliationData>
<OR>
                  <reconciliationAttribute
ucmdbAttributeName="primary_dns_name"
serviceDeskAttributeName="NetworkDNSName" priority="2" />
<AND priority="1" >
               <reconciliationAttribute ucmdbAttributeName="name"
serviceDeskAttributeName="NetworkName"
converterClassName="com.mercury.topaz.fcmdb.adapters.
serviceDeskAdapter.converter.PropertyValueConverterToUpperCase"/>
                <reconciliationAttribute ucmdbClassName="ip_
address" ucmdbAttributeName="name"
serviceDeskAttributeName="NetworkAddress" />
</AND>
</OR
   </reconciliationData>
```

In this example the reconciliation rule follows this format: (**node.primary_dns_name= NetworkDNSName OR (node.name= NetworkName and ip_address.name= NetworkAddress)**). Since the AND element takes a priority attribute of value `1`, the **(node.name= NetworkName and ip_address.name= NetworkAddress)** condition is checked first. If the condition is satisfied, the reconciliation is run. If not, the **.host_ dnsname= NetworkDNSName** condition is checked.

The additional sub-element of the **reconciliationClassConfiguration** element is **classConnectorConfiguration**. The **classConnectorConfiguration** element contains the configuration for a specific connector implementation for the current reconciliation CIT. This configuration should be wrapped by CDATA if it contains some special XML characters (for example, `&` replacing `&`).

Changing the Reconciliation Rule of a CIT

1. In **serviceDeskConfiguration.xml**, update the appropriate **`reconciliationData`** element with the new rule.
2. Call to the JMX to reload the adapter: **FCmdb Config Services > loadOrReloadCodeBaseForAdapterId**, using the appropriate customer ID and **`ServiceDeskAdapter`** adapter ID, or go to the Integration Points pane and reload the adapter from there. For details, see "Integration Point Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Reconciliation of a Host by IP Address or Name

To run reconciliation on a host by **ip_address** or **name**, place the following `ReconciliationData` element in the Adapter configuration file:

```
                        <reconciliationData>
                            <OR>
                                <reconciliationAttribute
priority="1" ucmdbClassName="ip_address" ucmdbAttributeName="ip_
address" serviceDeskAttributeName="NetworkAddress" />
                                <reconciliationAttribute
priority="2" ucmdbClassName="node" ucmdbAttributeName="name"
serviceDeskAttributeName="NetworkName"
converterClassName="com.mercury.topaz.fcmdb.adapters
.serviceDeskAdapter.converter.PropertyValueConverterToUpperCase" />
                            </OR>
                        </reconciliationData>
```

Global Configuration

The third section of the Adapter configuration file contains the global configuration for the specific connector implementation. This configuration, `globalConnectorConfig`, should be wrapped by CDATA if it contains some special XML characters (for example, `&` replacing `&`).

The useful fields of the Service Manager **`globalConnectorConfig`** element are as follows:

1. The **date_pattern** element contains the date adapter with which the Service Manager works.

   The default is `MM/dd/yy HH:mm:ss`.

   If the date adapter is wrong, an FTQL returns wrong date condition results.

2. The **time_zone** element defines the time zone of Service Manager. The default is the `UCMDB server time zone`.

   To check the Service Manager date adapter and time zone:

   a. **Service Manager version 7**: Access **Menu Navigation > System Administration > Base System Configuration > Miscellaneous >System Information Record**. Click the **Date Info** tab.

   b. **ServiceCenter version 6.1**: Access **Menu Navigation > Utilities > Administration > Information >System Information**. Click the **Date Info** tab.

3. The **max_query_length** element defines the maximal query length in a Service Manager Web service request. The default value is `1000000`.

4. The **name_space_uri** element defines the name space URI to connect to the Service Manager Web service. The default value is `http://servicecenter.peregrine.com/PWS`.

5. The **web_service_suffix** element defines the Service Manager Web service center URI suffix. The default value is `sc62server/ws`. It is used when the URL is created.

How to Deploy the Adapter – Typical Deployment

This section describes a typical deployment of the adapter.

This task includes the following steps:

1. "How to Deploy the ServiceDesk Adapter" below.
2. "How to Add an Attribute to the ServiceCenter/Service Manager CIT" on page 573.

How to Deploy the ServiceDesk Adapter

This section explains where to place the files needed for deployment.

This task includes the following steps:

- "Add a ServiceCenter/Service Manager External Data Source" on next page
- "Configure HP ServiceCenter 6.2" on next page
- "Configure HP Service Manager 7.0/7.1 " on page 571

- "Define data push jobs (optional)" on page 571
- "Run the jobs" on page 572
- "Select Classes for Federation" on page 572

1. Add a ServiceCenter/Service Manager External Data Source

a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

b. Click the **new integration point** button  to add an integration point.

- Click  , select the ServiceDesk Adapter that matches your version of Service Manager, and click **OK**.

Each out-of-the-box adapter comes predefined with the basic setup needed to perform integration with UCMDB. For information about changing these settings, see "Integration Studio Page" in the *HP Universal CMDB Data Flow Management Guide*.

- Enter the following information, and click **OK**:

Name	Description
CMDB State (Data Push)	The state of the source machine. Values are: ○ Actual ○ Authorized **Note:** This field is visible only on a UCMDB for which authorized state has been defined.
Credentials	Allows you to set credentials for integration points. For credential information, see "Supported Protocols" on page 82.
Hostname/IP	The name of the server on which HP Service Manager is running
Integration Name	The name you give to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
Port	The server port at which HP Service Manager is connected.

c. Click **Test connection** to verify the connectivity, and click **OK**.

d. Click **Next** and verify that the following message is displayed: **A connection has been successfully created**. If it does not, check the integration point parameters and try again.

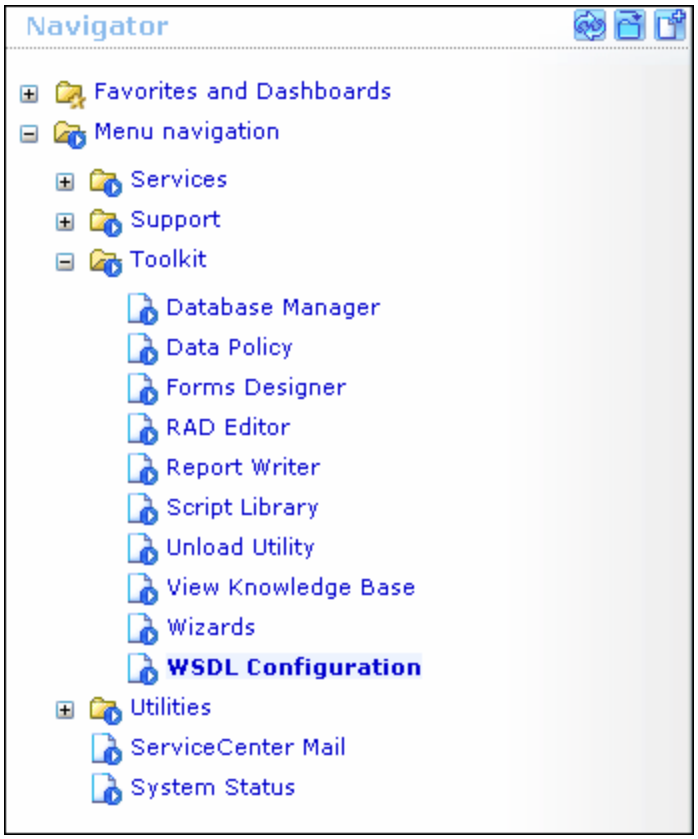
e. Continue with "Configure HP ServiceCenter 6.2" below or "Configure HP Service Manager 7.0/7.1 " on page 571.

2. Configure HP ServiceCenter 6.2

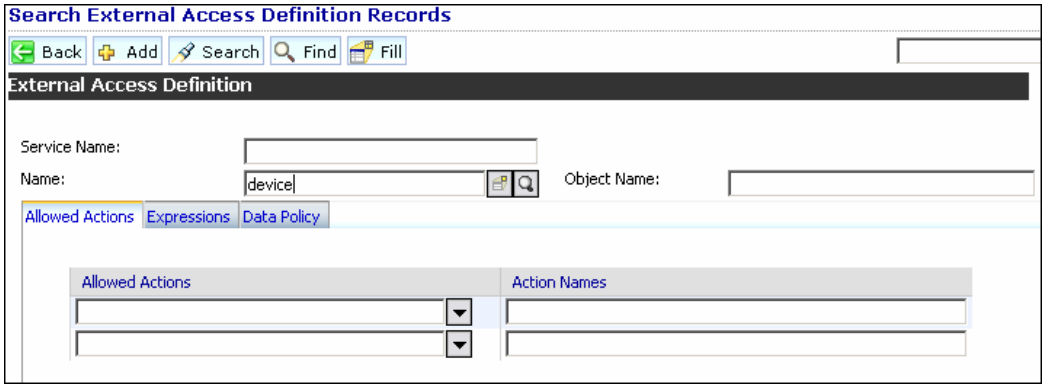
If you are connecting to HP ServiceCenter 6.2, perform the following procedure. If you are connecting to HP Service Manager 7.0/7.1, skip this step.

a. Open HP ServiceCenter, then the ServiceCenter client.

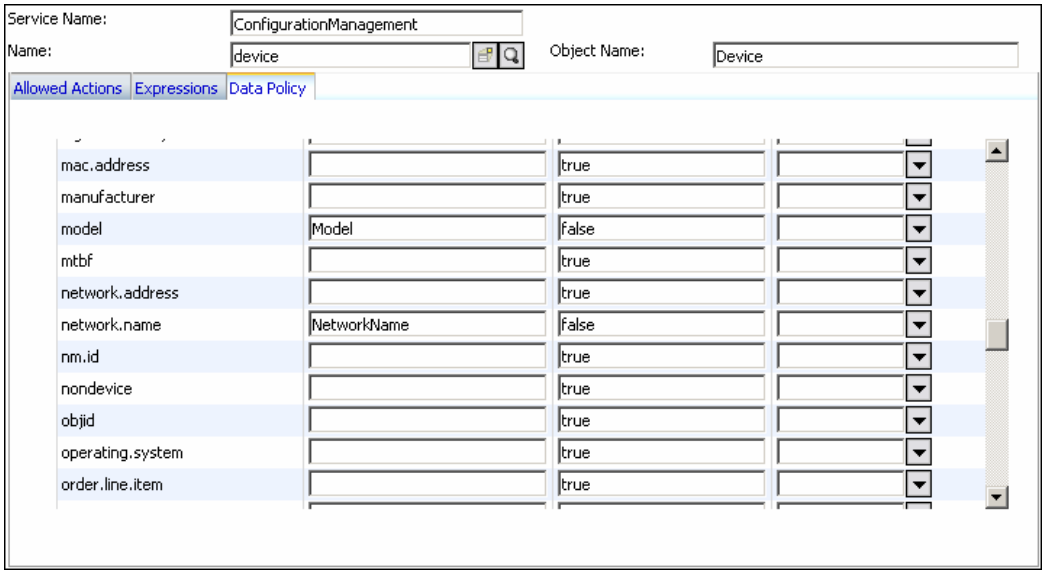
b. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu navigation > Toolkit**):



c. In the Name field, enter **device** and press **Enter**:



d. Select the **Data Policy** tab and ensure that the `network.name` attribute is not empty (its value should be **NetworkName**). Change the value to **false**. Save your changes.

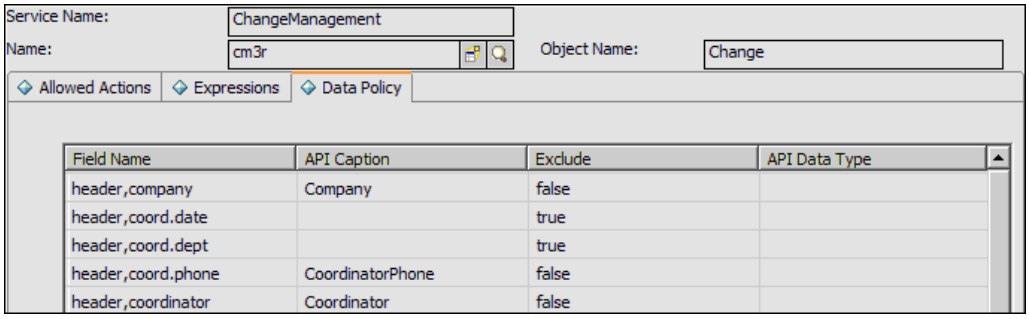


e. After saving, click the **Cancel** button.

f. In the Object Name field type **Change** and press **Enter**.

g. Select the Data Policy tab and ensure that:

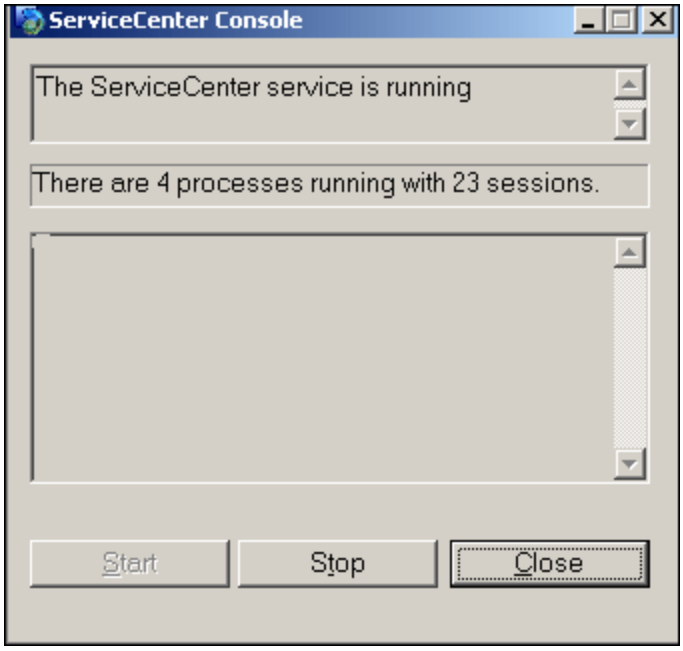
- The **header,coordinator** attribute is not empty (its value should be **Coordinator**). Change the value to **false**.



- The **header,orig.operator** attribute is not empty (its value should be **OpenedBy**). Change the value to **false**.

h. Save the changes.

i. Restart ServiceCenter: Select **Start > Programs > ServiceCenter 6.2 > Server > Console** to open the ServiceCenter Console.



j. Click **Stop** and then **Start**.

k. Continue with "Add an Attribute to the UCMDB Model" on page 579.

3. Configure HP Service Manager 7.0/7.1

If you are connecting to HP Service Manager 7.0/7.1, perform the following procedure. If you are connecting to HP ServiceCenter 6.2, skip this step.

a. Import the unload file relevant to the Service Manager version with which you are working: **ucmdbIntegration7_0x.unl** or **ucmdbIntegration7_1x.unl**. To do so, in Service Manager, click **Menu Navigation > Tailoring > Database Manager**.

- Right-click the detail button and select **Import/Load**.
- In the HP Service Manager File Load/Import page, click **Specify File** and browse to the following unload file:

  C:\hp\UCMDBServer\runtime\fcmdb\CodeBase\ServiceManagerAdapter7-1

  The file is loaded via the file browser.
- Enter the description in the **Import Description** box.
- Select **winnt** in the **File Type** list.
- Select a display option.
- Click **Load FG** to start loading.

b. Continue with "Add an Attribute to the UCMDB Model" on page 579.

4. Define data push jobs (optional)

Note: The Data Push flow is relevant for HP Service Manager version 7.1 and later only.

The Service Manager 7.1x-9.2x adapter comes out-of-the-box with the SM History-based Changes push job and the SM Topology Comparison RMI job, which use the queries described below.

- The SM History-based Changes push job uses the following predefined queries: **hostData**, **networkData**, **printerData**, **applicationData**, and **businessServiceData**.
- The SM Topology Comparison RMI job uses of the following predefined queries: **hostRelationsData**, **applicationRelationsData**, and **businessServiceRelationsData**.

For details about these queries, see "Predefined Queries for Data Push Jobs" on page 581.

Each of these jobs runs according to a default schedule setting.

You can also create additional jobs. To do this, select the Data Push tab to define data push jobs that uses the integration point you defined in "Add a ServiceCenter/Service Manager External Data Source" on page 568. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

5. ## Run the jobs

   a. Run the Changes Job, and then run the RMI job.

   b. Click the **Refresh Statistics** button  (**Data Flow Management > Integration Studio > Statistics tab**) to review the jobs' statistics. Compare the statistics to the TQLs by using the **Calculate Query Result Count** button  in the Modeling Studio.

   c. In Service Manager, verify that the CIs have been pushed correctly.

6. ## Select Classes for Federation

   The adapter contains the following predefined classes for federation: **request_for_change**, **problem**, and **incident**.

How to Add an Attribute to the ServiceCenter/Service Manager CIT

This section explains how to retrieve additional data from ServiceCenter or Service Manager by adding an attribute to the CIT.

This task includes the following steps:

- "Add an attribute to the UCMDB model" below
- "Export attributes from HP ServiceCenter by changing the configuration" below
- "Export attributes from HP Service Manager by changing the configuration" on next page
- "Modify the Adapter Configuration File" on page 576

1. ## Add an attribute to the UCMDB model

   Edit the Incident CIT to add the new attribute to UCMDB as follows:

   a. Navigate to **Modeling > CI Type Manager**.

   b. In the CI Types pane, select **IT Process Record > Incident**.

   c. Select the Attributes tab and add the new attribute.

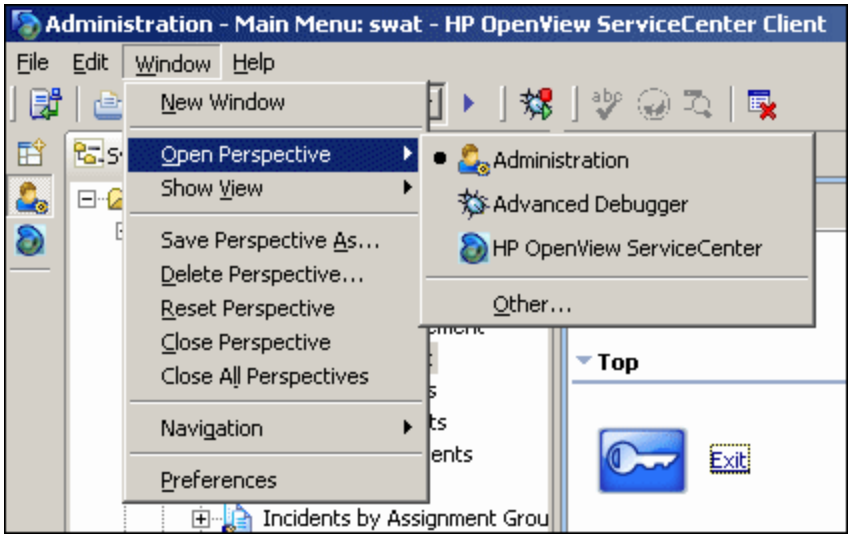
   d. Continue with "Export attributes from HP ServiceCenter by changing the configuration" below or "Export attributes from HP Service Manager by changing the configuration" on next page.

2. ## Export attributes from HP ServiceCenter by changing the configuration

   If you are connecting to HP ServiceCenter, perform the following procedure.

   a. In HP ServiceCenter, open the ServiceCenter client.

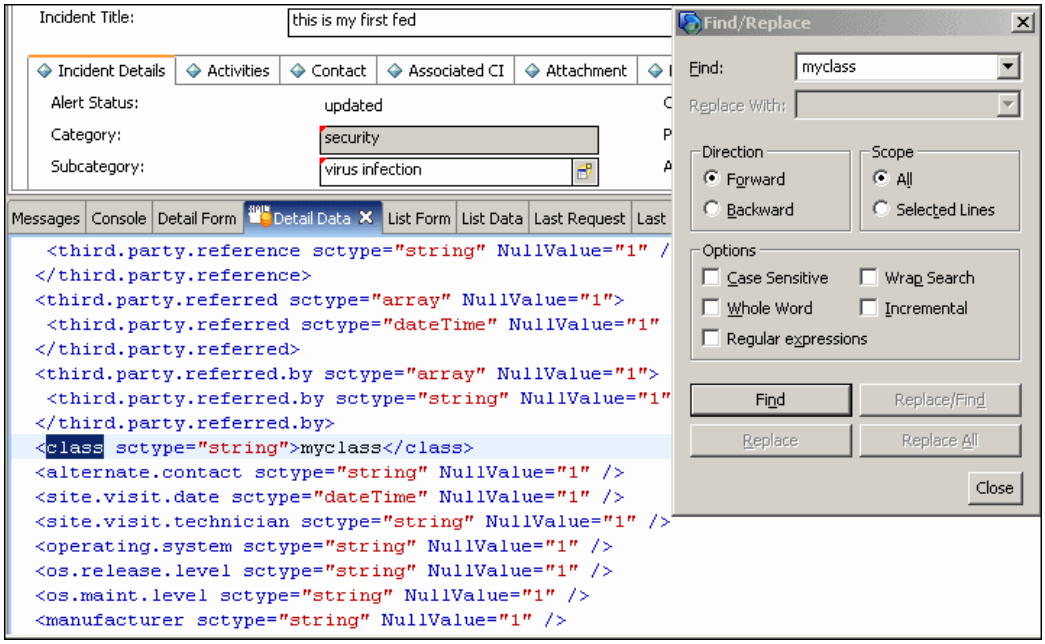
   b. Select **Window > Open Perspective > Administration**:



c. Select **Incident Management > All Open Incidents**, and select one of the incidents you created.

> **Note:** Verify that the value in the Class field is the one that you want to report to UCMDB.

d. Search for the value you entered in the Class field (that is, **myclass**), in the XML file displayed below. This is the CI name in ServiceContainer.



e. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu navigation > Toolkit**). Locate the Object Name field, enter **Incident** and press **Enter**.

f. Select the **Data Policy** tab. Enter a name for the CI mentioned in the XML file (that is, **class**). Change the value to **false**. Save your changes.

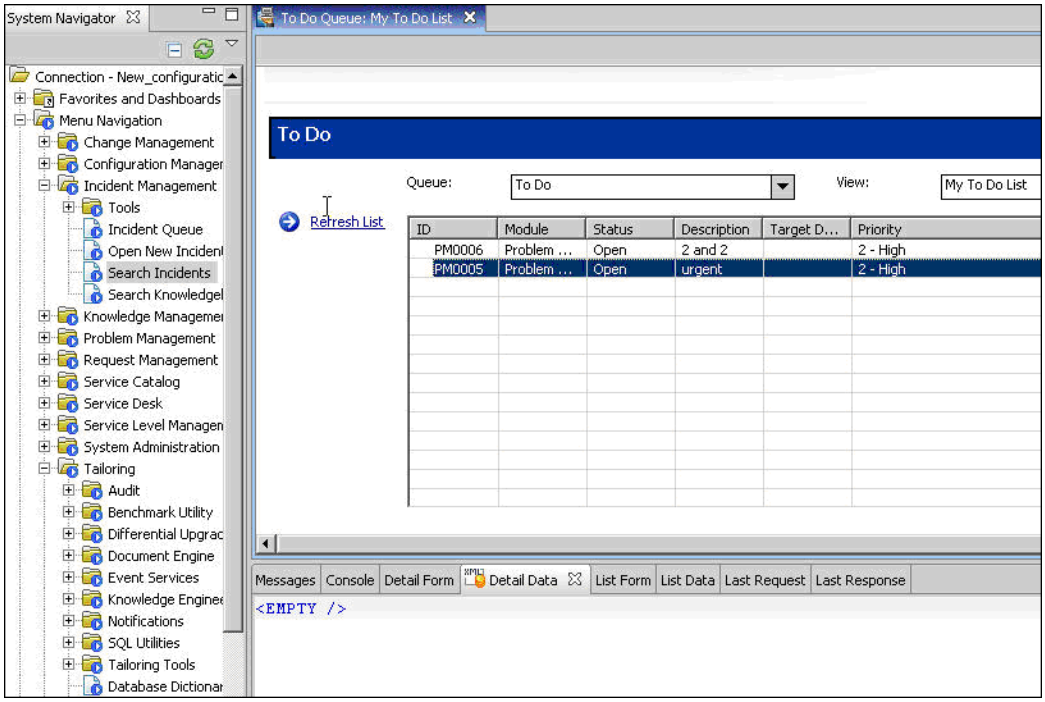
g. Restart ServiceCenter: Select **Start > Programs > ServiceCenter 6.2 > Server > Console** to open the ServiceCenter Console.

h. Click **Stop** and then **Start**.

3. Export attributes from HP Service Manager by changing the configuration

If you are connecting to HP Service Manager, perform the following procedure.

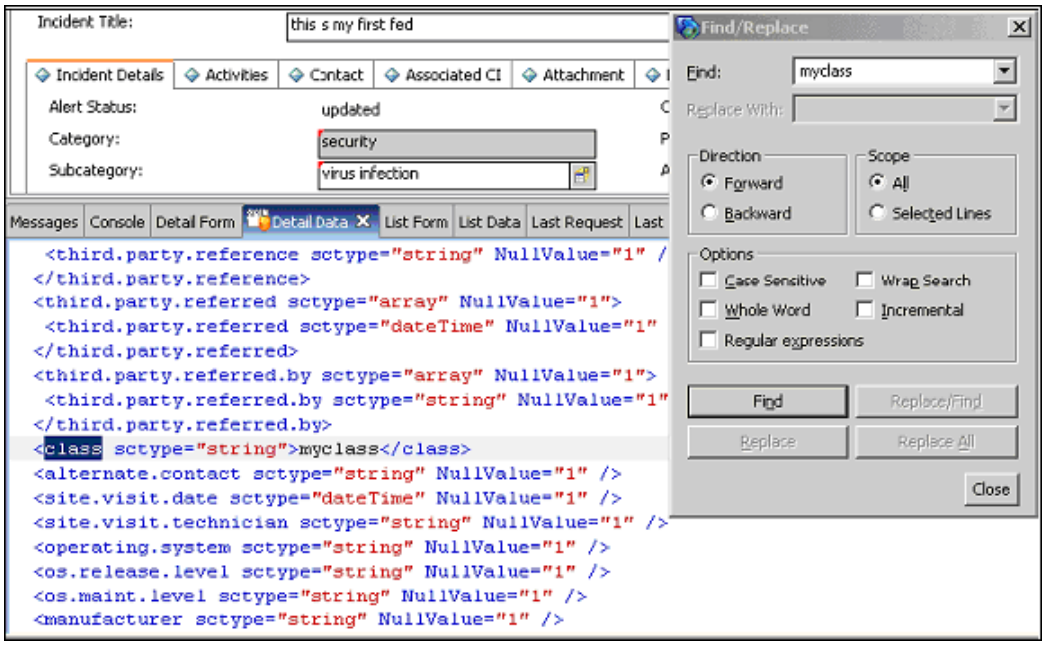
a. In the HP Service Manager client, restore the bottom right pane by clicking the **Restore** button. Click the **Detail Data** tab.



b. Open one of the incidents you created: Select **Incident Management > Search Incidents**. Click the search button (you can filter the fields to limit the search).

> **Note:** Verify that the value in the Class field is the one that you want to report to HP Universal CMDB.

c. Search for the value you entered in the Class field (that is, **myclass**), in the XML file displayed below. This is the CI name in Service Manager.



d. Display **WSDL Configuration** in the Navigator (**Main Menu > Menu Navigation > Tailoring**). Locate the Object Name field, enter **UcmdbIncident** and press **Enter**.

e. Select the **Data Policy** tab.

f. Select the **Fields** tab and ensure that the CI name mentioned in the XML file (that is, **class**) appears in the Field list with **ClassName** as its caption. If this attribute does not appear in the Field list, add it and save your changes.

g. Continue with "Modify the Adapter Configuration File" below.

4. Modify the Adapter Configuration File

Perform this procedure for all configurations.

a. Navigate to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapterthat corresponds to your version of Service Manager. Continue and select **Configuration Files > ServiceDeskConfiguration.xml**.

b. Edit the **ServiceDeskConfiguration.xml** file by navigating to **Data Flow Management > Adapter Management > ServiceManagerAdapter** (the one that corresponds to your version of Service Manager) **> Configuration Files > ServiceDeskConfiguration.xml**

c. Add the new attribute line under the Incident area: Locate the following marker:

```
<ucmdbClassConfiguration ucmdbClassName="it_incident">
<attributeMappings>
```

d. Add the following line:

```
<attributeMapping ucmdbAttributeName="incident_class"
ServiceDeskAttributeName="ClassName"/>
```

where:

- `ucmdbAttributeName="incident_class"` is the value defined in the CI Type Manager
- `ServiceDeskAttributeName="ClassName"` is the valued defined in ServiceCenter/Service Manager

e. Click **Save**.

How to Communicate with Service Manager over SSL

The following procedure explains how to open communication with Service Manager over SSL.

This task includes the following steps:

- "Add an SM Self-signed Certificate to the UCMDB Trusted Stores" below
- "Add the SM External Data Source Using Communication Over SSL " below

1. ## Add an SM Self-signed Certificate to the UCMDB Trusted Stores

   a. Copy the SM self-signed certificate to a directory. (To export SM self-signed certificates, refer to the Service Manager documentation).

   b. Locate the JRE security folder, by default located in: **C:\hp\UCMDB\UCMDBServer\bin\jre\lib**

   c. Back up the **cacerts** file by renaming it.

   d. Open a command line window and execute the following commands (to import the previously created or copied certificate):

   For HP Universal CMDB 8.0x:

   ```
   cd C:\hp\UCMDB\UCMDBServer \jre\bin"
   keytool.exe -import -keystore
   C:\hp\UCMDB\UCMDBServer\j2f\JRE\lib\security\cacerts" -
   trustcacerts -file
   <full path to SM self-signed certificate>
   ```

   For HP Universal CMDB 9.00 or later:

   ```
   cd C:\hp\UCMDB\UCMDBServer\bin\jre\bin
   keytool.exe -import -keystore
   C:\hp\UCMDB\UCMDBServer\bin\jre\lib\security\cacerts -
   trustcacerts -file
   <full path to SM self-signed certificate>
   ```

   e. Restart the UCMDB service.

2. ## Add the SM External Data Source Using Communication Over SSL

   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

   b. Define an integration point using the following parameters: In the new integration point dialog box, choose the **ServiceDeskAdapter** for your version of ServiceCenter or Service Manager, and enter the user name, password, and URL. The URL field should contain: **https://<SM server name>:13443/sc62server/ws**.

   For details, see "New Integration Point/Edit Integration Point Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

How to Add a New Attribute to an Existing CI Type

Perform the following steps to add a new attribute to an existing CI type.

This task includes the following steps:

- "Add an Attribute to the UCMDB Model" below
- "Add the Attribute to the Layout of the TQL Query" below
- "Map the Attribute in the SM Adapter Configuration" below
- "Map the Field in the Service Manager Web Service" below

1. **Add an Attribute to the UCMDB Model**
   a. Navigate to **Modeling > CI Type Manager**.
   b. Select the CI type to which you want to add the attribute.
   c. Select the Attributes tab and add the new attribute.
2. **Add the Attribute to the Layout of the TQL Query**
   a. Navigate to **Modeling > Modeling Studio**.
   b. Select the query that contains the CI type you want to change (located in the **Integration\SM Sync** folder).
   c. Right-click the node of the CI type you are changing and select **Query Node Properties**.
3. **Map the Attribute in the SM Adapter Configuration**
   a. Navigate to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.
   b. Select Configuration Files, and choose the xslt file that contains the CI type you changed.
   c. Add the attribute at the file.device XML tag or at the concrete file XML tag of the type (depends on the Service ManagerWeb Service).
4. **Map the Field in the Service Manager Web Service**
   For details, refer to the Service Manager documentation.

How to Add a New CI Type

Perform the following steps to add a new CI type to the UCMDB class model.

This task includes the following steps:

1. **Add the CI Type to the UCMDB Class Model**
   a. Navigate to **Modeling > CI Type Manager**.
   b. Add the new CI type and its valid relations.
2. **Define a TQL Query for Synchronizing the CI Type**
   a. Navigate to **Modeling > Modeling Studio**.
   b. In the **Integration\SM Sync** folder, create a new query.

      The new TQL query should include the new CI type (which should be labeled as `Root`) and all the related CIs that are connected to the root node for the additional data. For example: in the **hostData** query, IpAddress and Interface are the additional data of the node.

      The TQL query should also contain the layout that you want to synchronize.
3. **Map the Attribute in the SM Adapter Configuration**
   a. Navigate to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.
   b. Select Configuration Files, and choose the xslt file that contains the CI type you changed.
   c. Add the attribute at the file.device XML tag or at the concrete file XML tag of the type (depends on the Service Manager Web Service).
4. **Map the CI Type in the SM Adapter Configuration**
   a. Navigate to **Data Flow Management > Adapter Management** and select the ServiceManagerAdapter that corresponds to your version of Service Manager.
   b. Select Configuration Files.
   c. Create a new xslt file for the new CI type and map all the attributes and related CIs to it.
   d. Open **smSyncConfFile.xml** and add a mapping between the new TQL query and the new xslt file.
5. **Create and Map the Field in the Service Manager Web Service**

For details, refer to the Service Manager documentation.

6. **Update the Data Push Job**
   a. Navigate to **Data Flow Management > Integration Studio**.
   b. Configure the Data Push job to include the new TQL query.

Predefined Queries for Data Push Jobs

The following TQL queries (located in the Modeling Studio in the **Integration\SM Sync** folder) are provided out-of-the-box if you use the ServiceCenter/Service Manager adapters when you create an integration point.

This section includes:

- "Queries for Data Push Changes Job (SM History-based Changes job)" below
- "Queries for a Data Push RMI job (SM Topology Comparison RMI job)" below

Queries for Data Push Changes Job (SM History-based Changes job)

These queries are used to create a job of type Changes (for pushing CIs):

- **hostData** – use to push nodes. Pushed data includes nodes whose NodeRole attribute is either empty, or contains desktop, server or virtualized_system. Nodes are identified either by their interface or IP address. Information also includes the location of the nodes (building, floor, and room). Due to limitations of the Changes flow, the location information is saved using an enrichment in the Calculated Location attribute.
- **networkData** – use to push nodes that are not pushed with the **hostData** query. This query is similar to **hostData**, except that it pushes nodes whose NodeRole attribute is not empty and does not contain the following strings: `desktop`, `server`, `virtualized_system`, or `printer`.
- **printerData** – use to push printers (network printers). This query is similar to **networkData**, except that it does push nodes where the NodeRole attribute contains the string `printer`.
- **applicationData** – use to push Business Applications.
- **businessServiceData** – used to push Business Services.

For details, see "Integration Jobs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Note:

- Select the Allow Delete check box if you want your Data Push job to send deletes of CIs & Links to Service Manager.
- The Changes flow is required for integration with Service Manager because it creates a single CI out of a topology, which matches the Service Manager specification.

Queries for a Data Push RMI job (SM Topology Comparison RMI job)

These queries are used to create a job of type RMI (for pushing Relations):

- **hostRelationsData** – use to push Layer2 (Physical) connections between pairs of nodes through their interfaces.
- **applicationRelationsData** – use to push logical relations between Business Applications to other Business Applications and nodes.
- **businessServiceRelationsData** – use to push logical relations between Business Services to other Business Services, applications and nodes.

For details, see "Integration Jobs Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Flow and Configuration

The ServiceCenter/Service Manager adapter receives data and a TQL definition from the Data Push engine, transforms it into a SOAP call for each instance of the TQL query's results, and sends the SOAP requests to Service Manager.

The transformation between the UCMDB class model to the Service Manager class model is done by an XSLT engine.

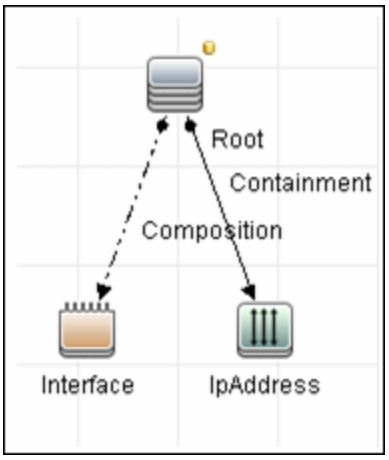
This section also includes:

- "Parse the TQL Definition" below
- "XSLT Transformation" on page 586

Parse the TQL Definition

The TQL definition must have one Root node (in which case it will be considered a CI synchronization TQL) or several Root links (in which case it will be considered a Relations synchronization TQL).

Example of an out-of-the-box TQL query for synchronizing a node CI type:



To XML

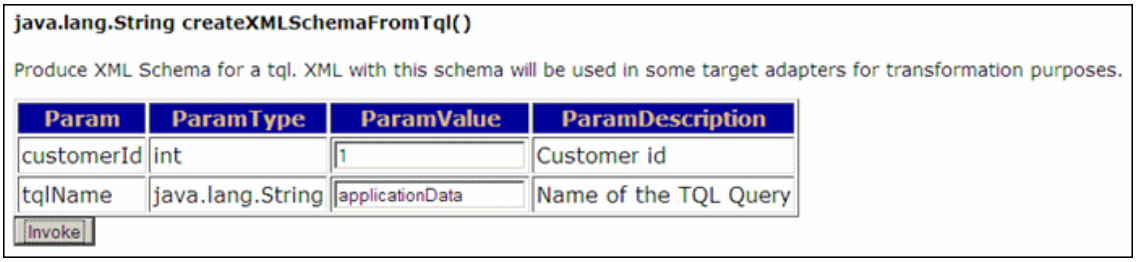
The result of the TQL query is divided into instances according to the Root node/links, and each instance is given an XML representation.

XML Schema

Each TQL query is automatically assigned a schema according to the structure of the TQL adapter and the layout attributes chosen.

Example of an XML schema for a TQL query:

This example displays the XML schema for a TQL query using a UCMDB JMX located at **http://[cmdb_machine]:8080/jmx-console/HtmlAdaptor, service=FCmdb Config Services, createXMLSchemaFromTql(**



XML schema for a **networkData** TQL query example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <<xs:element name="node">
    <xs:complexType>
      <xs:sequence>
       <xs:element name="ip_addresss" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="ip_address"
minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:attribute name="friendlyType"
type="xs:string"/>
                                    <xs:attribute name="id"
type="xs:string"/>
                                    <xs:attribute name="ip_netmask"
type="xs:string"/>
                                    <xs:attribute name="name"
type="xs:string"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="interfaces" minOccurs="0"
maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="interface" minOccurs="0"
maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:attribute name="friendlyType"
```

```
type="xs:string"/>
                                        <xs:attribute name="id"
type="xs:string"/>
                                        <xs:attribute name="mac_address"
type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
            </xs:sequence>
            <xs:attribute name="calculated_location"
type="xs:string"/>
            <xs:attribute name="default_gateway_ip_address"
type="xs:string"/>
            <xs:attribute name="discovered_os_name" type="xs:string"/>
            <xs:attribute name="discovered_os_version"
type="xs:string"/>
            <xs:attribute name="friendlyType" type="xs:string"/>
            <xs:attribute name="global_id" type="xs:string"/>
            <xs:attribute name="id" type="xs:string"/>
            <xs:attribute name="node_role" type="xs:string"/>
            <xs:attribute name="primary_dns_name" type="xs:string"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Example of XML for a **networkData** query:

```
<node customer_id="1" discovered_os_name="windows 2010"
    discovered_os_version="build45-2a" friendlyType="Net Device"
    global_id="bdef388c1b1b3db863ce442a96b54e53"
id="bdef388c1b1b3db863ce442a96b54e53"
    default_gateway_ip_address="1.2.3.4"
    calculated_location="Room:234 Floor:2 Building:M54"
node_role="&lt;Values&gt;&lt;Value&gt;firewall&lt;
/Value&gt;&lt;/Values&gt;" primary_dns_name="myDNS.com">
    <ip_addresss direction="outgoing" linkType="Containment">
        <ip_address customer_id="1" friendlyType="IpAddress"
             id="91757d9d45f166437c1864e931f59e16" ip_
address="16.59.64.1"/>
        <ip_address customer_id="1" friendlyType="IpAddress"
            id="f91bf4c40b06e460b51af2178181843d" ip_
address="16.59.66.1"/>
    </ip_addresss>
</node>
```

XSLT Transformation

Mapping a TQL name to XSLT

To map between the TQL names and the XSL files, navigate to **Data Flow Management > Adapter Management > ServiceManagerAdapter** (the one that corresponds to your version of Service Manager) **> Configuration Files > smSyncConfFile.xml**.

Example of XML for configuring a **hostData** query:

The file inludes the names of the Service Manager requests for each operation (create, update, and delete).

```
<tql name="hostData" xslFile="host_data.xslt">
   <!-- this is host->ip,interface,sm_host tql -->
   <request type="Create" name="CreateucmdbComputerRequest"/>
   <request type="Update" name="UpdateucmdbComputerRequest"/>
   <request type="Delete" name="DeleteucmdbComputerRequest"/>
 </tql>
```

The **smSyncConfFile.xml** file must be updated when you add a new TQL query that will be synchronized with Service Manager.

Result after transformation

This sample shows the results after **host_data1.xslt** is executed on the original XML file.

```
<UpdateucmdbNetworkRequest>
            <model>
                <keys/>
                <instance>
                    <file.device>

                        <UCMDBId>bdef388c1b1b3db863ce442a96b54e53</UCMDBId>
                        <CustomerId>1</CustomerId>
                        <Subtype>firewall</Subtype>
                        <Building>M54</Building>
                        <Floor>2</Floor>
                        <Room>234</Room>
                        <DefaultGateway>1.2.3.4</DefaultGateway>
                        <OS>windows 2010</OS>
                        <DNSName>myDNS.com</DNSName>
                    </file.device>
                    <file.networkcomponents>
                        <OSVersion>build45-2a</OSVersion>
                        <addlIPAddr>
                            <addlIPAddr>

                                <AddlIPAddress>16.59.64.1</AddlIPAddress>
                                <AddlSubnet/>
                            </addlIPAddr>
                            <addlIPAddr>
```

```
                                <AddlIPAddress>16.59.66.1</AddlIPAddress>
                                <AddlSubnet/>
                            </addlIPAddr>
                        </addlIPAddr>
                    </file.networkcomponents>
                </instance>
            </model>
        </UpdateucmdbNetworkRequest>
```

XSLT references

XSLT is a standard language for transforming XML documents into other XML documents. The adapter uses the built-in Java 1.5 Xalan XSLT 1.0 transformer. For details about XSLT see:

http://www.w3.org/TR/1999/REC-xslt-19991116

http://www.w3schools.com/xsl/

http://www.zvon.org/xxl/XSLTutorial/Output/index.html

Reuse of XSLT parts

In addition to the standard XSLT specifications, the adapter? supports the use of an XSLT preprocessor that scans XSL files for comments such as **<!--import:[file_name]-->** in the XSLT, and replaces them with the contents of **[file_name]**.

Service Manager WSDL

Tools such as SoapUI or SoapSonar can be used to view the WSDL files.

Service Manager Web Services are dynamic and can be modified. For details on how to edit or add new Service Manager Web Services, refer to the Service Manager documentation.

Service Manager Result SOAP request

For details on how to enable printing of SOAP requests, see "Logs" on next page.

Using Mapping Tools

An automatic tool (such as Mapforce) can be used to create XSLT mappings between the CMDB XML schema and the Service Manager XML schema.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for the ServiceCenter/Service Manager adapter.

Changes Flow Limitations

- A query should contain one CI that is labeled as `Root` or one or more relations that are labeled as `Root_<postfix%gt;`.

  The root node is the main CI that is synchronized, and the other nodes are the contained CIs of the main CI. For example, when synchronizing Nodes, the query node of (Node) will be labeled as `Root` and the host resources will not be root.
- The TQL graph must not contain cycles.
- The TQL query must only contain the Root CI, and optionally CIs that are directly connected to it.
- A query that is used to synchronize relations should have cardinality 1...* and OR condition between them.
- Any conditions must reside on the Root CI only.
- If you want to synchronize only specific Roots from a TQL query, you must configure the required condition on these Roots, and then, configure the same condition in the TQL that synchronize the relationships that are linked to the Roots.
- Compound relations are not supported.
- Subgraphs are not supported.
- if one of the TQL queries that are used for synchronization (including layout changes) is edited, the changes will not be synchronized until a full data push job has been manually run. Results from a previous synchronization will not be deleted from the Service Manager server.
- Changes to NodeRole only will not be detected and will not update CI for the next Data Push job.

Logs

Use the **fcmdb.adapters.log** file to troubleshoot the Service Desk adapter (located in the **UCMDBServer\runtime\log** folder).

To view the complete SOAP request and response in addition to other information, use the **fcmdb.properties** file to change the adapter's log level to debug: **log4j.category.fcmdb.adapters=debug,fcmdb.adapters**.

Do not forget to change the log level back to **error** when you are finished debugging. For example, if the **fcmdb.adapters.log** of an Service Manager integration names SM01, for each single CI sent the log will show:

```
DEBUG - SM01 >> Source CI tree is: (The XML as outputted by the ucmdb
goes here)
INFO - SM01 >> ======== start run soap message
INFO - SM01 >> ========== create urs required time = 0
DEBUG - SM01 >> Run message: (The XML Send after Xslt Transformation
```

```
goes here)
DEBUG - SM01 >> Response message: (The XML response goes here)
INFO - SM01 >> ======== stop run soap message. The required time = 390
```

In multi-threaded push flows the thread name indicates the chunk number and thread number:

```
[SM01_pushObjectWorkerThread-<ChunkID>::<ThreadID>]
```

Actual State

To troubleshoot the Actual State flow, use a SOAP testing tool such as SoapUI or SoapSonar to run a SOAP request similar to this:

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:types="http://schemas.hp.com/ucmdb/1/types"%gt;
  <soap:Body>
    <types:getAllCIProperties>
      <types:ID>17868889fd660853e16a474e10df5de3</types:ID>
    </types:getAllCIProperties>
  </soap:Body>
</soap:Envelope>
```

You will obtain a response similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header />
<soapenv:Body>
<types:getAllCIPropertiesResponse
xmlns:types="http://schemas.hp.com/ucmdb/1/types">
<types:CI id="17868889fd660853e16a474e10df5de3" type="Windows">
<types:prop type="string">
<types:name>Host Name</types:name>
<types:value>LABM2AM209</types:value>
</types:prop>
<types:prop type="string">
<types:name>Host Operating System</types:name>
<types:value>Windows 2003 Server Enterprise Edition </types:value>
</types:prop>
<types:prop type="string">
<types:name>Host Vendor</types:name>
<types:value>Microsoft Windows</types:value>
</types:prop>
<types:prop type="string">
<types:name>Host DNS Name</types:name>
<types:value>labm2am209.devlab.ad</types:value>
</types:prop>
<types:prop type="string">
```

```
<types:name>Asset Tag</types:name>
<types:value>GB8718DS72___</types:value>
</types:prop>
<types:complexProp className="IP" size="1">
<types:item>
<types:prop type="string">
<types:name>IP Address</types:name>
<types:value>16.59.56.161</types:value>
</types:prop>
<types:prop type="string">
<types:name>IP Network Mask</types:name>
<types:value />
</types:prop>
</types:item>
</types:complexProp>
...
</types:CI>
</types:getAllCIPropertiesResponse>
</soapenv:Body>
</soapenv:Envelope>
```

If errors occur, review the following files for exceptions:

- **C:\hp\UCMDB\UCMDBServer\runtime\log\error.log**
- **C:\hp\UCMDB\UCMDBServer\runtime\log\cmdb.operation.log**

Chapter 47

HP Systems Insight Manager (HP SIM) Integration

This chapter includes:

Overview

HP Universal CMDB (UCMDB) can discover data center infrastructure information stored in an HP Systems Insight Manager (HP SIM) system. Integration involves synchronizing devices, topology, and the hierarchy of a data center infrastructure in the UCMDB database (CMDB). This enables change management and impact analysis across all business services mapped in UCMDB, from an infrastructure point of view.

UCMDB initiates discovery on the HP SIM server through Web service calls. Synchronized configuration items (CIs) include nodes such as Windows, and UNIX servers, network devices, printers, clusters, cellular/partitioned systems, blade enclosures, and racks. Some server components, for example, CPU, are also synchronized. The integration also synchronizes relationships between blade servers and blade enclosures, virtual machines, physical servers, and so on. The synchronization uses an XML-based mapping that dynamically changes synchronized CIs and attributes without requiring a code change.

For details on nodes and attributes in HP SIM, refer to the Database tables section of the *HP SIM Technical Reference* guide.

Supported Versions

This integration solution supports HP SIM versions 5.1, 5.2, 5.3, 6.0, 6.1, 6.2, and 6.3.

HP SIM Integration Mechanism

DFM uses the HP SIM Web service API to retrieve node information from the HP SIM database. DFM also enables you to specify extended attributes that should be retrieved for each node.

HP SIM represents hosts (blade enclosures, racks, servers, and so on) as Nodes; UCMDB has separate CITs for each such host. To represent hosts correctly in UCMDB, a two-level mapping is used, to enable integration customization without code changes. This makes the integration completely customizable and dynamic.

For details on jobs, see "Discovery Control Panel – Advanced Mode Workflow" in the *HP Universal CMDB Data Flow Management Guide*.

This section describes the two levels of mapping:

- "HP SIM Node to HP UCMDB Node Mapping" on next page
- "Node Attribute to CI Type and CI Attribute Mapping" on page 597

HP SIM Node to HP UCMDB Node Mapping

All IP-enabled systems are represented as **Nodes** in HP SIM and each node has attributes (for example, operating device type and operating system name) that can be used to classify nodes as specific CITs in UCMDB. The first level of mapping involves setting parameters on the **SIM Integration** job. This job includes **HostCitIdentifierAttributes** and **HostCitIdentifierMap** parameters that are used for the mapping:

- **HostCitIdentifierAttributes**. This attribute specifies the names of HP SIM Node attributes that are used for the mapping. This parameter uses the **DeviceType** and **OSName** out-of-the-box Node attributes. The parameter accepts comma-separated node attribute names, is case sensitive, and expects each node attribute name to be enclosed in single quotes.
- **HostCitIdentifierMap**. This attribute specifies the mapping between values of the above HP SIM Node attributes and corresponding UCMDB CITs. This parameter accepts a comma-separated list of value pairs, where each value pair takes the following format:

```
'node attribute value':'UCMDB CI Type'
```

Both attributes are case-sensitive and must be enclosed in single quotes. Each Node-attribute value is one possible value of one or more Node attribute names specified in the **HostCitIdentifierAttributes** parameter. Each UCMDB CIT is the name (not the display name) of the UCMDB CIT to which this value maps.

This parameter has out-of-the-box mappings as follows:

HP SIM Node Attribute	UCMDB CIT
'AIX'	'unix'
'Complex'	'complex'
'Embedded'	'management_processor'
'Enclosure'	'enclosure'
'HPUX'	'unix'
'Hypervisor'	'unix'
'LINUX'	'unix'
'MgmtProc'	'management_processor'
'Printer'	'netprinter'
'Rack'	'rack'
'Server'	'node'
'Solaris'	'unix'
'Switch'	'switch'
'WINNT'	'nt'

HP SIM Node Attribute	UCMDB CIT
'Workstation'	'node'

Example mapping based on the above settings:

- If the **DeviceType** attribute of a node has the value **Switch**, in UCMDB the node is represented as a **Switch** CIT.
- If the **OSName** attribute of a node has the value **WINNT**, in UCMDB the node is represented as an **NT** CIT (Display name: **Windows**).

The DFM script parses these mapping parameters from left to right and does not stop on success, so the rightmost match is considered final. This means that if a node has **DeviceName = Server** and **OSName = HPUX**, the rightmost match is **OSName** with value **HPUX**. The resulting CIT for this node in UCMDB is **unix** because **HPUX** maps to **unix**.

Node Attribute to CI Type and CI Attribute Mapping

Once the nodes are mapped to CITs using DFM job parameters as described in "HP SIM Node to HP UCMDB Node Mapping" on page 595, individual node attributes (including extended node attributes) are mapped to corresponding attributes (or CITs, as appropriate) using a generic UCMDB integration framework. The framework uses an XML file in which source and target CIT and attribute names are specified.

A sample XML mapping file (**SIM_To_UCMDB_Sample_MappingFile.xml**) that includes all node CITs mapped in the"HP SIM Node to HP UCMDB Node Mapping" on page 595 section is included in the **SIM_Integration** package. The sample file includes host resources (for example, CPU, Disk) and relationship mapping information, to build relationships between various nodes (for example, Blade Enclosure to server, virtual machine host to guest, and so on).

Using this framework, you can map additional CITs without any code changes. For example, to map HBAs, add a new section to the XML file. Define the node attributes that identify an HBA and its attributes. Relationships between HBAs and HOSTs are also required.

How to Discover HP SIM Data Center Infrastructure

This task describes how to discover data center infrastructure information stored in an HP Systems Insight Manager (HP SIM) system.

This task includes the following steps:

- "Prerequisites" below
- "Perform setup on the Probe machine" on next page
- "Enable chunking - optional" on page 599
- Run the job:
  - "Run the job - UCMDB 9.04 and later" on page 599
  - "Run the job - UCMDB 9.03 and 9.02" on page 600

1. **Prerequisites**

> **Important**: If you set up an HTTPS connection to connect to the SIM WebService API (that is, an SSL enabled HTTP connection), the **SIM Integration** job performs **no validation** of any certificates presented by the HP SIM server. The job trusts any certificate issued by the HP SIM server and uses it for SSL enabled communication.

The following additional requirements must be satisfied for the mapping file to be valid for HP SIM (for details on the mapping files, see "HP SIM Integration Mechanism" on page 594):

- Verify that source and target are **HP SIM** and **HP UCMDB** respectively.
- Verify that attribute names specified in the **HostCitIdentifierAttributes** parameter are

included as attributes of each host CIT in the XML file.

That is, the **OSName** and **DeviceType** attributes must be included for each **host_node** (Computer), **chassis** (Chassis), **netprinter** (Net Printer), **switch** (Switch), **nt** (Windows), **unix** (UNIX), **hp_complex** (Complex), and **management_processor** (Management Processor) CIT.

- Verify that default attributes (that is, non-extended attributes) of a node have a **Node.** prefix in the mapping file.

  That is, you should specify attributes such as **OSName**, **DeviceType**, and **IPAddress** as **Node.OSName**, **Node.DeviceType**, and **Node.IPAddress**.

- Verify that each Node CIT has the following attribute mapping to enable the generation of the **host_key** attribute:

```
<target_attribute name="host_key" datatype="StrProp" >
    <map type="direct" source_attribute="host_key" />
</target_attribute>
```

> **Note:** The **host_key** attribute is the primary key attribute on Node and derived CITs. Since HP SIM uses a different type of key attribute, the XML definition for the **host_key** attribute is included in the mapping file, to enable generation of the **host_key** primary key attribute.

- Verify that the IP Address mapping section has the following attribute to enable automatic population of the IP domain attribute:

```
<target_attribute name="ip_domain" datatype="StrProp">
    <map type="direct" source_attribute="ip_domain" />
</target_attribute>
```

> **Note:** For details on the list of HP SIM nodes and attributes, refer to the HP SIM documentation.

2. ## Perform setup on the Probe machine

   a. Copy **mxpartnerlib.jar** from this directory:

   C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\hpsim

   to this directory:

   C:\hp\UCMDB\DataFlowProbe\content\lib

   b. Open **C:\hp\UCMDB\DataFlowProbe\bin\WrapperEnv.conf** for editing.

   c. Comment out line ~51 with a hash sign (**#**) at the beginning so that it looks as follows:

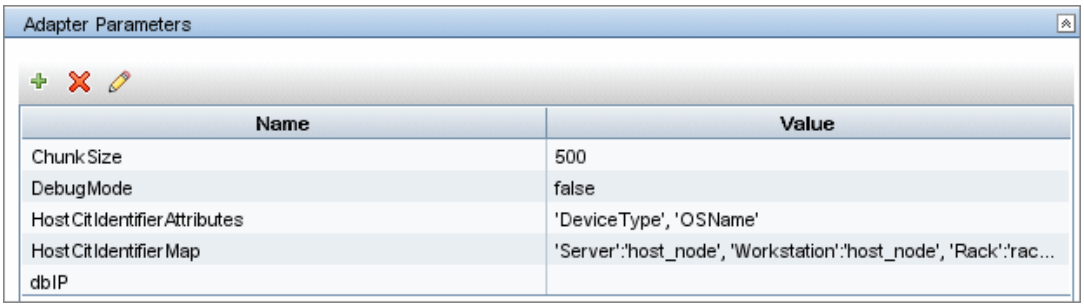
   ```
   #set.SYSTINET_CLASSES=%lib%/webservice;.........
   ```

   d. Save and close the file.

e. Restart the Probe.

3. ## Enable chunking - optional

If the HP SIM server being discovered contains or manages a large number of nodes (more than 1,000), you should consider enabling chunking (**Data Flow Management > Adapter Management > select an adapter > Adapter Management tab > Adapter Parameters pane**):



a. To reduce load on the SIM server, if necessary, you can set the **ChunkSize** parameter in the **SIM Integration** adapter to a lower value than the default **500**.

b. Populate the **dbIP** parameter in the **SIM Integration** adapter with the IP address of the HP SIM CMS database.

c. Populate the **SIM Database ...** fields in the HP SIM protocol with connection details for the HP SIM CMS database.

> **Note:** HP SIM CMS database details (except for the password) are located in the **Systems Insight Manager\config\database.props** file on the HP SIM server.

4. ## Run the job - UCMDB 9.04 and later

> **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

In DFM, in the Integration Studio, create a new integration point.

a. Provide a name and description for the integration point.

b. Under **Integration Properties > Adapter**, select the **Systems Insight Manager** adapter.

c. Complete the **dbIP** field with the IP address of the HP SIM CMS database.

d. Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.

e. Under **Adapter Properties > Trigger CI instance** select:

i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or

ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

> **Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

f. Save the integration point.

g. Run the job.

5. **Run the job - UCMDB 9.03 and 9.02**

> **Note:** To enable inclusion in a UCMDB spiral discovery schedule, the job is split into two. The **SIM WebService Ports** job triggers on all IpAddress CIs in the CMDB and looks for port 50001—the port at which HP SIM listens for Web service queries. The **SIM Integration by WebService** job triggers on results from the SIM WebService Ports job and retrieves data.

Prerequisite - Set up protocol credentials

a. Set up the HP SIM Protocol credentials (Data Flow Management **> Data Flow Probe Setup > Domains and Probes > <domain name> Credentials > HP SIM Protocol**).

For credential information, see "Supported Protocols" on page 82.

> **Note:** By default, the following fields are required: **Port Number**, **SIM WebService Protocol**, **User Name**, and **User Password**. The **SIM Database ...** fields are required if the **dbIP** parameter on the discovery job is populated. For details, see "Enable chunking - optional" on previous page.

Run the following jobs in the following order:

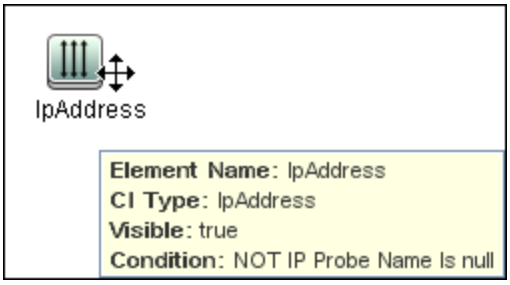
b. Run the **Range IPs by ICMP** job to discover the IP address of the HP SIM server.

c. Run the **SIM WebService Ports** job to discover the Web service ports on the HP SIM server. This job triggers on all **IpAddress** CIs in the CMDB and looks for port 50001 (the port at which HP SIM listens for Web service queries). For job details, see "SIM WebService Ports Job" on next page.

d. Run the **SIM Integration by WebServices** job to discover HP SIM infrastructure. This job triggers on results from the **SIM WebService Ports** job and retrieves data. For job details, see "SIM Integration by WebServices Job" on page 602.

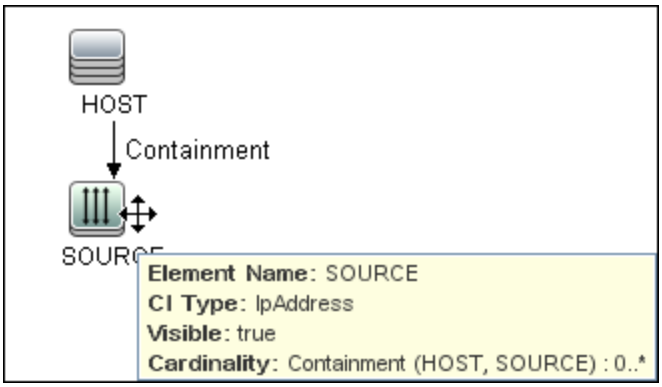
SIM WebService Ports Job

Trigger Query



Adapter

- **Input query:**

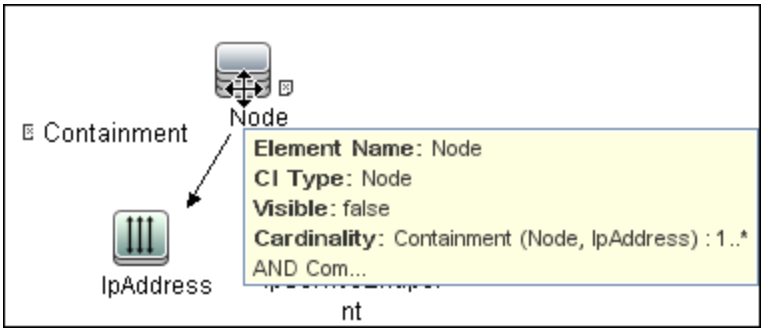


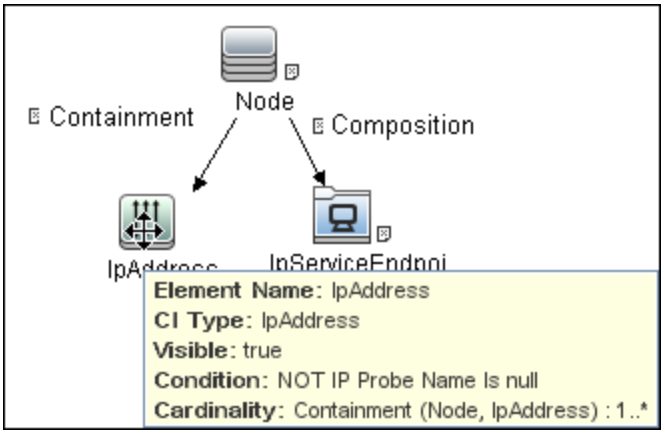
Discovered CITs

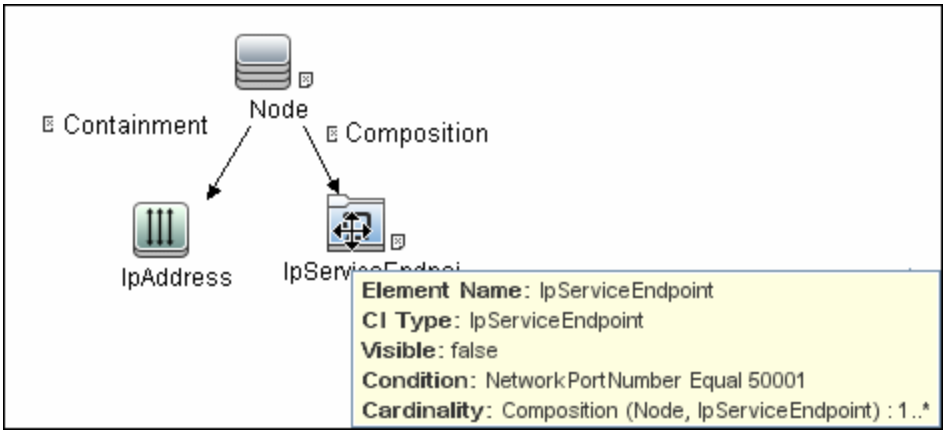
- **Composition**
- **Containment**
- **IpAddress**
- **IpServiceEndpoint**
- **Node**
- **Usage**

SIM Integration by WebServices Job

Trigger Query







Discovered CITs

- **Chassis**
- **Composition**
- **Computer**
- **Containment**

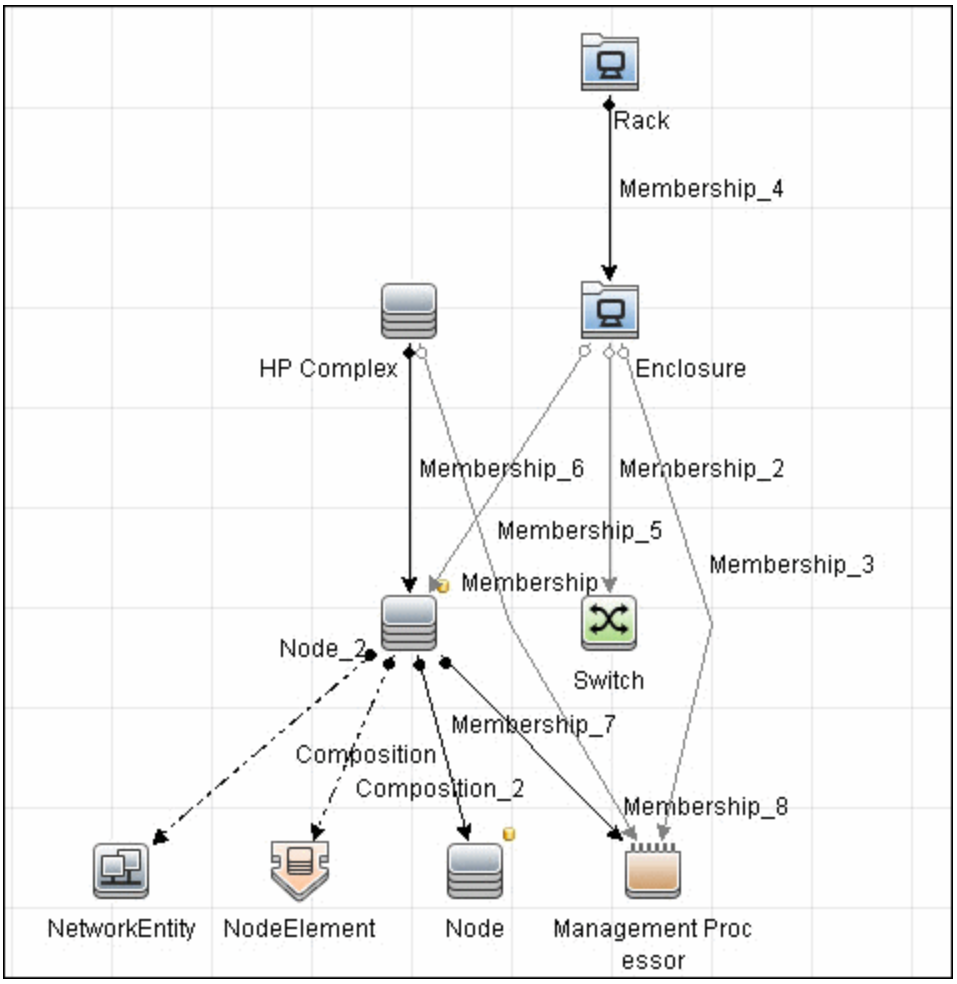
- **Cpu**
- **Dependency**
- **Enclosure**
- **HP Complex**
- **Interface**
- **IpAddress**
- **LogicalVolume**
- **Management Processor**
- **Membership**
- **Net Printer**
- **Node**
- **Process**
- **Rack**
- **Switch**

Instance Views

The package includes two adapter views that show all nodes and resources retrieved from HP SIM, as well as relationships between these nodes.

This section includes the following topics:

Host Infrastructure View

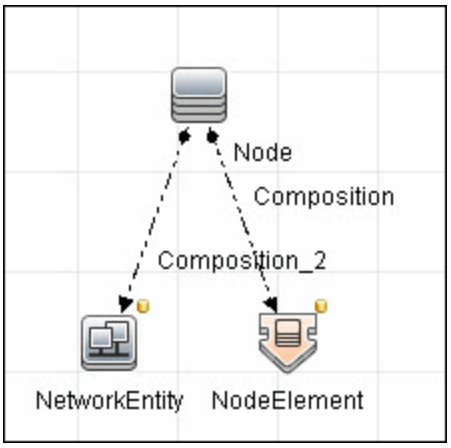


This view shows relationships between Chassis, Blade Enclosures, Servers, Workstations, Virtual Machine hosts to guests, and so on. This view also shows the interdependence between various nodes in an environment, to enable change management and correlation.

You can use this view, for example, to identify all the servers housed within a specific blade enclosure and all virtual machines running on servers within this blade enclosure. This enables analysis of the impact of shutting down a blade enclosure (say, for a firmware upgrade) on virtual machines. If UCMDB knows of services provided by these virtual machines and which business

service these services are part of, it becomes possible to analyze the impact of a blade enclosure outage all the way to a business service.

Hosts and Resources from HP SIM



This view shows Node CIs retrieved from HP SIM with associated HostResource and NetworkResource CIs also retrieved from HP SIM.

Troubleshooting and Limitations

This section describes troubleshooting and limatations for HP SIM integration.

Note: The following limitation only applies when (a) using UCMDB Version 9.00-9.03, **and** (b) the user manually increased the out-of-the-box value in the pattern execution option **maximum threads** to greater than 1.

Limitation: If there are multiple HP SIM servers in the environment and this integration is used to integrate with all of them, you should create a new integration job for each HP SIM server and schedule them to run separately. This is because the integration uses XML files to process results from HP SIM, and running the integration against multiple HP SIM servers simultaneously causes the XML files to be overwritten (because the file name is static).

Chapter 48

IDS Scheer ARIS Integration

This chapter includes:

Overview

UCMDB integration with IDS Scheer ARIS IT Architect (ARIS) involves synchronizing business services/processes and Enterprise Architecture (EA) information from ARIS to the UCMDB database. This enables end-to-end Change Management and Impact Analysis from the IT infrastructure (at the data center level) to the business service/process level.

The integration involves a UCMDB initiated pull of information from an XML export generated by ARIS. Synchronized configuration items (CIs) include Business Service, Business Process, Business Process Step, Ownership information and Business Application (software). The integration requires manual reconciliation of business application instances in UCMDB.

ARIS_Integration.zip, contains the views, scripts, adapters, and jobs for the IDS Scheer ARIS Integration.

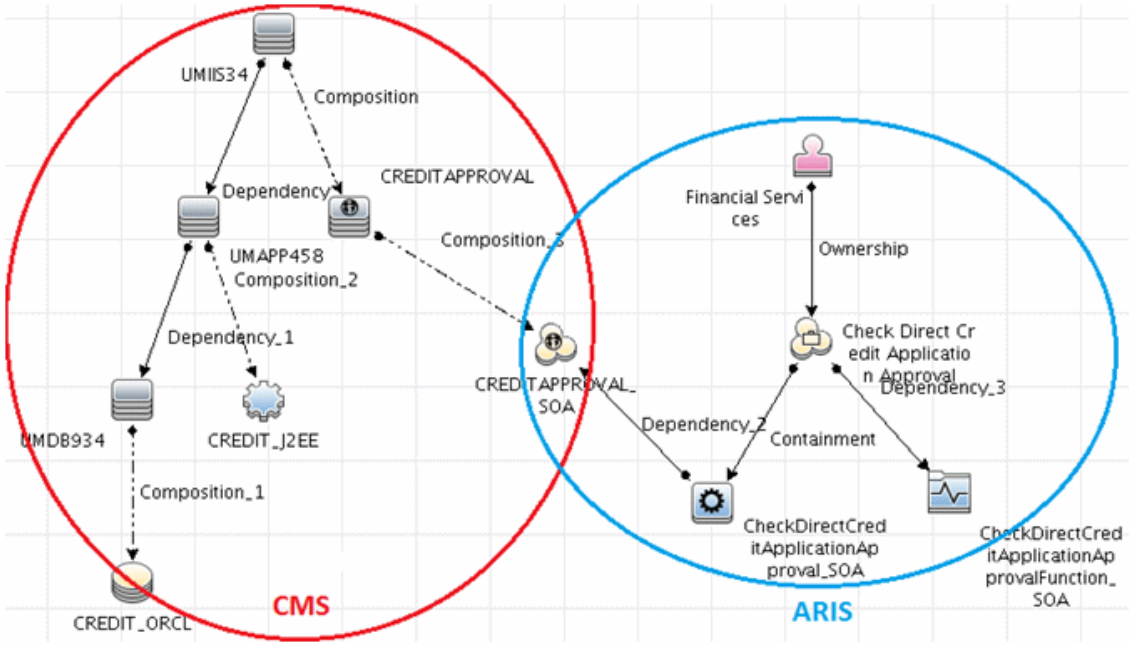
Supported Versions

This integration supports ARIS IT Architect version 7.1.

Topology

The following image is a sample topology showing relationships between the IT infrastructure (data center layer) and Business Processes/Services.

Note: For a list of discovered CITs, see "Discovered CITs" on page 614.



How to Run the ARIS Integration Job

This task includes the steps to run the ARIS integration job, to integrate the IDS Scheer ARIS IT Architect CIs into UCMDB.

This task includes the following steps:

- "Export the ARIS model to an XML file" below
- "Set up the ARIS-UCMDB mapping" below
- Run the job:
  - "Run the job - UCMDB 9.04 and later" on page 612
  - "Run the job - UCMDB 9.03 and 9.02" on page 613

1. ### Export the ARIS model to an XML file

   This integration solution uses an XML output file generated by ARIS. It is recommended to export the ARIS model to a minimal XML file for use by the UCMDB integration job.

   When exporting the data:

   - The output XML file should NOT be compressed.
   - The language of the output file must be the same as the language used for UCMDB.
   - Configure settings as follows:
     - Assignments: No assignments
     - Connections: n connections, with a connection level of 1
     - Select to perform a minimum export
     - Options to export users and groups and group structures should NOT be selected.

     Note: Save the exported file to a location accessible to the Data Flow Probe.

   For more details on exporting XML files in ARIS, contact your IDS Scheer support representative or ARIS IT Architect documentation.

2. ### Set up the ARIS-UCMDB mapping

   Data flow is initiated by UCMDB reading the XML file generated by ARIS. The integration job reads the data in this file and creates CIs.

   A user configurable mapping file (also in XML format) may be used to customize mapping of:

   - ARIS Object Types to UCMDB CI types
   - ARIS links to UCMDB relationships

   This mapping XML file, **ARIS_To_UCMDB.xml**, is located in the following folder:

   ```
   <UCMDB installation>\UCMDB\DataFlowProbe\runtime\
   probeManager\discoveryResources\TQLExport\ARIS\data
   ```

To set up the ARIS Object Type - UCMDB CI Type mapping:

Note: These mapping instructions are followed by an illustrated example.

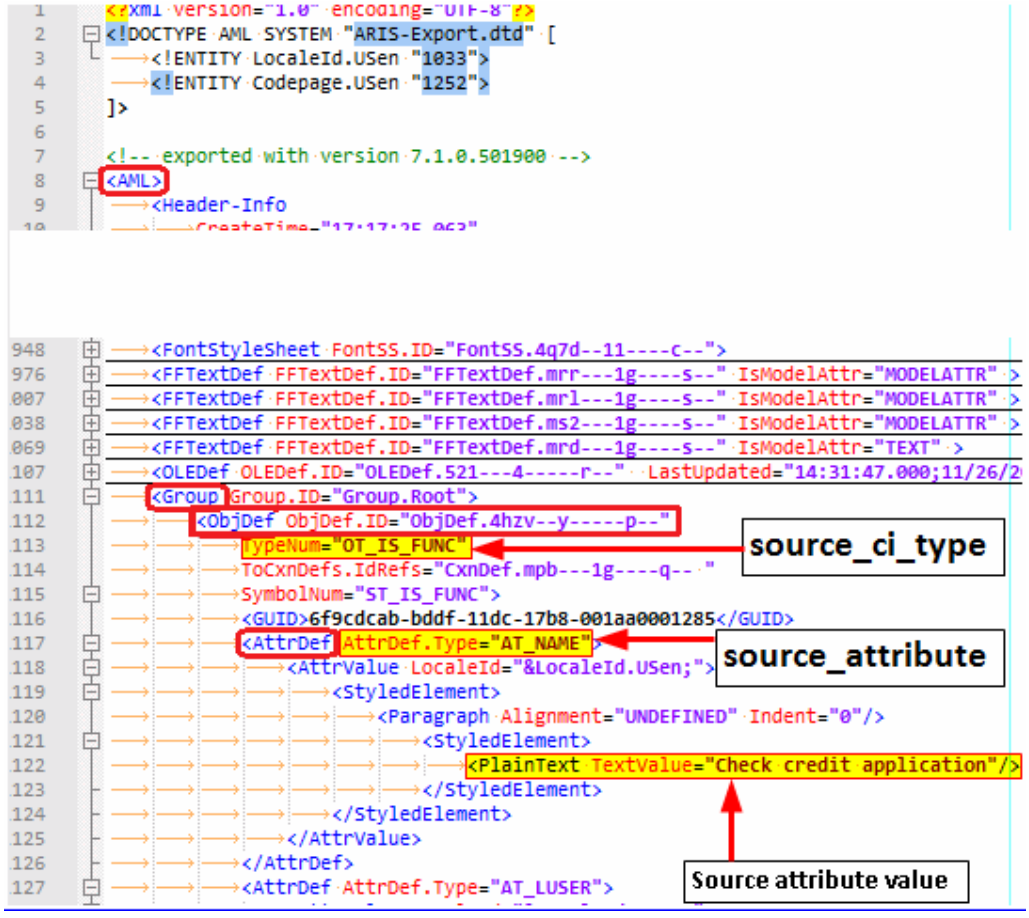
a. For each ARIS object type that you want to map, in the exported ARIS XML file (the **source** XML) locate the relevant **ObjDef** tag, and note the **TypeNum** and **AttrDef.Type** values.

b. In the mapping file, **ARIS_To_UCMDB.xml**, locate the **<targetcis>** section and enter these values into the **source_CI_type namesource_attribute** attributes respectively.

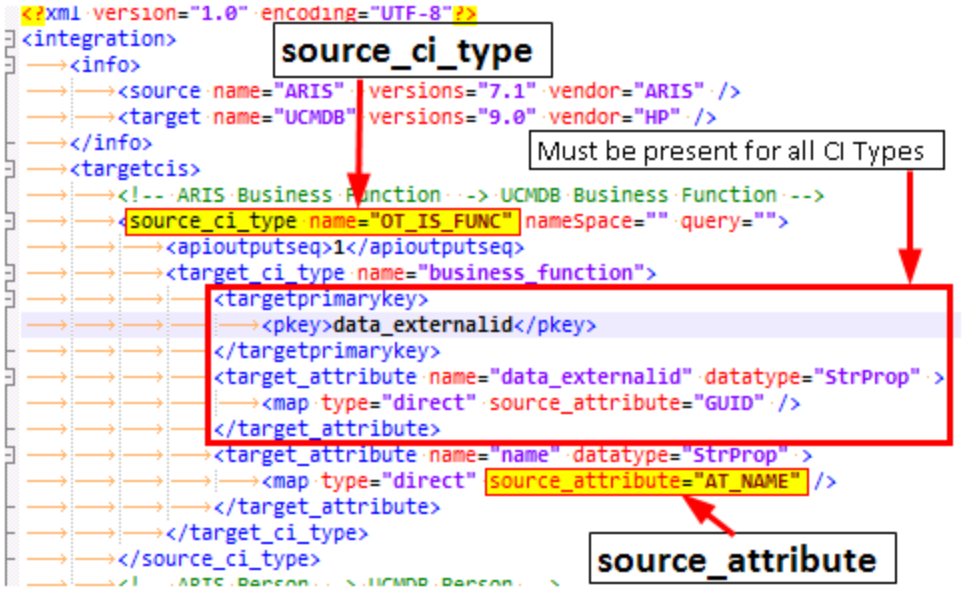
Example:

In the following image of the source XML file, the object, **ObjDef.4hzv--y-----p--**, has the following attribute values:

- TypeNum = **OT_IS_FUNC**
- AttrDef.Type = **AT_NAME**



These values are entered in the mapping file's **source_CI_type name** and **source_attribute** attributes, as illustrated below:



Note: The section marked as **Must be present for all CI Types** must exist for ALL CI type mappings defined in the mapping file. This section populates the unique object ID used by ARIS in the "data_externalid" attribute of the UCMDB CI type.

To set up the ARIS Link - UCMDB Relationship mapping:

Note: These mapping instructions are followed by an illustrated example.

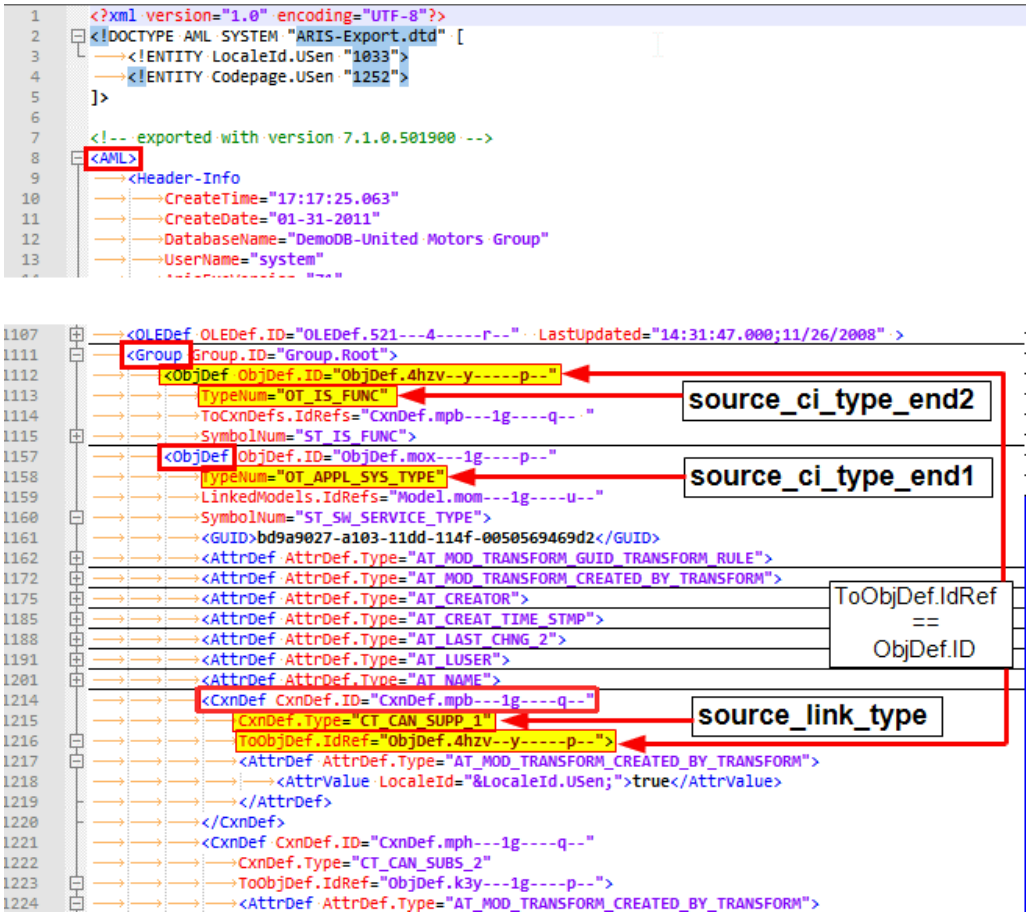
c. For each ARIS link that you want to map, note the following values in the source XML file:
   - Locate the relevant **CxnDef** tag and note the **CxnDef.Type** attribute.
   - Locate the CxnDef tag's parent, **ObjDef**. Note the **TypeNum** value under this ObjDef.
   - Under CxnDef, note the **ToObjDef.IDRef** attribute, and search for an ObjDef tag with the identical value. Then, under this ObjDef, note the **TypeNum** attribute.

d. In the mapping file, **ARIS_To_UCMDB.xml**, locate the **<targetrelations>** section and enter the source link's values as follows:
   - For **source_link_type**, enter the CxnDef.Type attribute
   - For **source_ci_type_end1**, enter the TypeNum value of the CxnDef tag's parent.
   - For **source_ci_type_end2**, enter the TypeNum value of the ObjDef that is equivalent to the ToObjDef.IDRef

Example:

In the following image of the source XML file, the link, **CxnDefn CxnDef.ID=CxnDef.mpb---1g----q--**, has the following attribute values:

- CxnDef.Type = **CT_CAN_SUPP_1**
- CxnDef's parent's **TypeNum** attribute = **OT_APPL_SYS_TYPE**
- ToObjDef.IDRef = **ObjDef.4hzv--y-----p--**. The equivalent ObjDef, **ObjDef.4hzv--y-----p--**, was found in line 1112, and its TypeNum attribute is **OT_IS_FUNC.**



These values are entered in the mapping file's **<link>** tag, in the **source_link_type**, **source_ci_type_end1**, and **source_ci_type_end2** attributes respectively, as illustrated below:

```
34      <target_attribute name="name" datatype="StrProp
35          <map type="direct" source_attribute="AT_NAM
36      </target_attribute>
37    </target_ci_type>
38  </source_ci_type>
39</targetcis>
40
41<targetrelations>
42  <!-- ARIS link -> UCMDB relationship -->
43  <link source_link_type="CT_CAN_SUPP_1"
44      target_link_type="usage"
45      nameSpace=""
46      mode="update_else_insert"
47      source_ci_type_end1="OT_APPL_SYS_TYPE"
48      source_ci_type_end2="OT_IS_FUNC"
49      query="">
50    <apioutputseq>4</apioutputseq>
51    <target_ci_type_end1 name="business_application"/>
52    <target_ci_type_end2 name="business_function"/>
53    <targetprimarykey>
54      <pkey>name</pkey>
55    </targetprimarykey>
56  </link>
57</targetrelations>
58
59</integration>
```

3. Run the job - UCMDB 9.04 and later

In DFM, in the Integration Studio, create a new integration point.

a. Provide a name and description for the integration point.

b. Under **Integration Properties > Adapter**, select the **Software AG ARIS** adapter.

c. Fill in the value for **ARIS_XML_file**. Set the value as the path to the XML file containing the exported ARIS data. See "Export the ARIS model to an XML file" on page 608.

d. Copy the DTD file, **ARIS-Export.dtd** from **<ARIS server>\Program Files\ARIS7.1\aml**to the directory where you saved the exported ARIS XML.

e. Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.

f. Under **Adapter Properties > Trigger CI instance** select:

i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or

ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

> **Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

g. Save the integration point.

h. Run the job.

> **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

4. ### Run the job - UCMDB 9.03 and 9.02

Set up the integration

To import data from the XML file into UCMDB:

a. In UCMDB, select the **Import CIs from ARIS** job, and override the default value of the **ARIS_XML_file** parameter as follows:

- Select to override the default value.
- Set the new value as the path to the XML file containing the exported ARIS data (see step "Export the ARIS model to an XML file" on page 608 above).

For user interface details, see the description about the Parameters pane in the *HP Universal CMDB Data Flow Management Guide*.

b. Copy the DTD file, **ARIS-Export.dtd** from **<ARIS server>\Program Files\ARIS7.1\aml**to the directory where you saved the exported ARIS XML.

Run the job

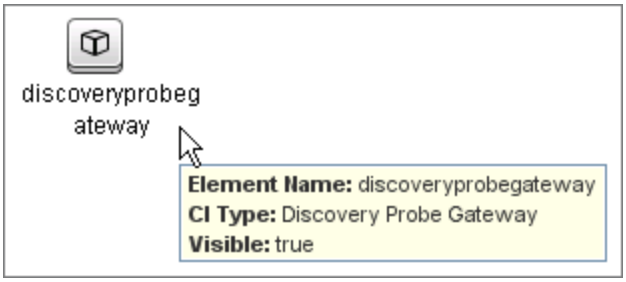
a. Activate the **Import CIs from ARIS** job. For job details, see "Import CIs from ARIS Job" below.

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

Import CIs from ARIS Job

Trigger Query

- **Trigger CI:** Probe
- **Trigger query:**



Adapter

- **Input query:** There is no input query for this job.

Discovered CITs

The UCMDB-ARIS integration discovers the following CITs:

- **Business Process**
- **Business Activity**
- **Business Function**
- **Business Application**

Note: To view the topology, see "Topology" on page 607.

Chapter 49

Import from Excel Workbook Discovery

This chapter includes:

Overview

This document describes the usage and functionality of the XLS_Import discovery package developed for importing UCMDB topology from a Microsoft Excel (*.xls, *.xlsx) file.

Supported Versions

This discovery supports

- Microsoft Excel files, versions 97, 2000, XP, and 2003 (*.xls)
- Office Open XML format for Excel 2007 (*.xlsx)

Topology

The following image displays the topology of the Import from Excel discovery.

Note: The topology discovered by the Import from Excel Workbook job relies on import file content, so only root objects are enumerated as discovered CITs. For a list of discovered CITs, see "Discovered CITs" on page 628.

How to Import Data from Excel Workbook

This task describes how to run the Import from Excel discovery. The Import from Excel Workbook job imports data from the Probe's file system (or accessible network share), so no credentials are required.

Note: The Import from Excel Sample job is similar to the Import from Excel Workbook job. It differs only by reference to the sample import file.

This task includes the following steps:

1. **Prequisite- Set up the Import file in Excel**

   For details on setting up the import file, see " How to Set Up Import File in Excel" on page 619.

2. **Prerequisite - Set up permissions**

   Give the Data Flow Probe read permissions on the location on the file system where the import files are stored.

3. **Prerequisite - Modify the Probe class path**

   a. Edit the following file: **C:\hp**UCMDB**\DataFlowProbe\bin\WrapperEnv.conf**.

   b. Locate the **Environment global vars** section and add the following line to the end of the section:

   ```
   set.probeManager=%runtime%/probeManager
   ```

   c. Locate the **Environment Discovery Path** section and add the following line:

   ```
   set.POI_CLASSES=%probemanager%/discoveryResources/geronimo-stax-
   api_1.0_spec-1.0.jar;%probemanager%/discoveryResources/poi-3.7-
   beta1-20100620.jar;%probemanager%/discoveryResources/poi-ooxml-
   3.7-beta1-20100620.jar;%probemanager%/discoveryResources/poi-
   ooxml-schemas-3.7-beta1-
   20100620.jar;%probemanager%/discoveryResources/xmlbeans-
   2.3.0.jar
   ```

   d. Do one of the following, according to your environment:

   - Modify the **COMMON_CLASSPATH** variable and insert the **%POI_CLASSES%** reference somewhere before the **%NNM_CLASSES%** reference. For example:

     ```
     set.COMMON_CLASSPATH=%POI_CLASSES%;%conf%;%XML_
     CLASSES%;%JYTHON_CLASSES%;%NNM_CLASSES%;...
     ```

   - Add the following line directly after set.COMMON_CLASSPATH=....:

     ```
     set.COMMON_CLASSPATH=%POI_CLASSES%;%COMMON_CLASSPATH%
     ```

   e. Restart the Probe.

4. ## Run the discovery

How you run this job depends on the software version being used.

UCMDB 9.04 (and later), with CP10

In DFM, in the Integration Studio, create a new integration point.

- Provide a name and description for the integration point.
- Under **Integration Properties > Adapter**, select the **Import from Excel Workbook** adapter.
- Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.
- Under **Adapter Properties > Trigger CI instance** select:
  - **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI; or
  - **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

    Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.
- Save the Integration Point.
- Run the job.

Note: For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

UCMDB 9.03 (and earlier)

- Activate the **Import from Excel Workbook** job.

How to Set Up Import File in Excel

This section describes how to define an import file. The following topology is created:

- Two hosts
- Two IPs contained by each host
- Network (the IPs mentioned above are members of the network)
- An application with a corresponding process running on the host

This task includes the following steps:

1. **Prerequisite**

   Open a new Excel file and name it **tutorial.xls**.

2. **Add a CI type**

   Double-click the **Sheet1** tab and rename it with the desired CI type. For this tutorial, use the name **node**.

   Note:
   - Only use the CI type name, not the display name.
   - Type names are case sensitive.

3. **Create Comment sheets - optional**

   You can create Comment sheets that will not be imported into UCMDB, but that can be used to describe the data contained in the imported document.

   Double-click one of the Sheet tabs and rename it **#Comment sheet**.

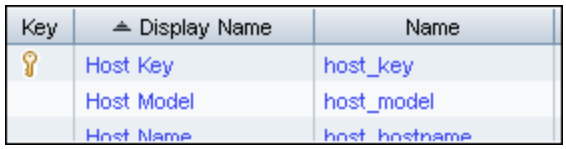
   

Note: Comment sheet names must begin with the **#** sign.

4. **Define CI key attributes**

Depending on the CIT, to store a CI in UCMDB, you must specify CI key attributes or attributes that participate in reconciliation rules. The names of the imported attributes can be defined as the column headings.

Our **node** object only has one key attribute - **host_key**.



Key	Display Name	Name
🔑	Host Key	host_key
	Host Model	host_model
	Host Name	host_hostname

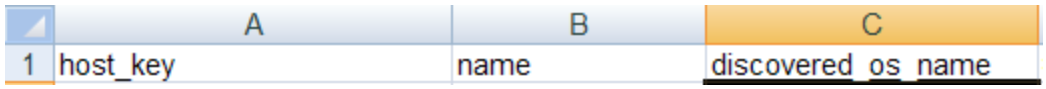
To import a node CI into UCMDB, you must set the **host_key** attribute. You may do this by one of the following methods:

- Set **host_key** in the view **<IP address> <Domain>**. (For example: 192.168.100.100 DefaultDomain.) This is enough to import a node CI into UCMDB.
- Set **host_key** as the lowest MAC address of the attached network interface. This is not enough to import a node CI into UCMDB. You must also configure the following attributes:
  i. Set **host_iscomplete** to **true**.
  ii. Set values for the node attributes that allow the node to be identified by the reconciliation rule. **Note:** The node reconciliation rule also allows identification of the nodes linked to this node IP address or network interface CIs. If you prefer to identify nodes by linked CIs, you must ensure the Excel document also has the imported IP/Network interface CIs, and the relationships between node CIs and IP/ network interface CIs.

Note:

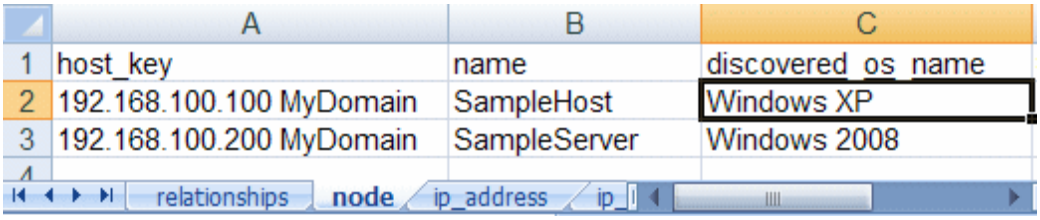
- The column headings must be attribute names, not display names.
- Attribute names are case sensitive.

You can show the node name and the operating system.



	A	B	C
1	host_key	name	discovered_os_name

a. Define two nodes.

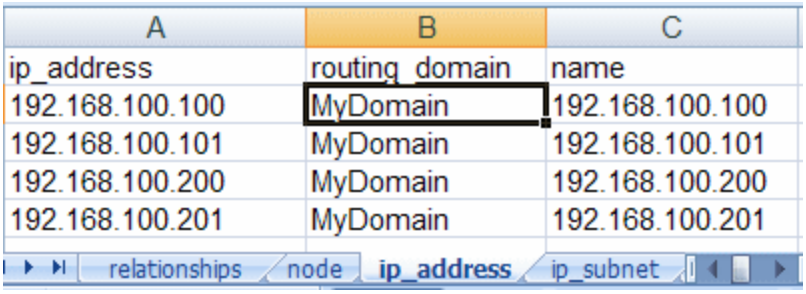


	A	B	C
1	host_key	name	discovered_os_name
2	192.168.100.100 MyDomain	SampleHost	Windows XP
3	192.168.100.200 MyDomain	SampleServer	Windows 2008
4			

relationships | node | ip_address | ip_

> **Note:** Each row in the sheet (except the first one) represents a single CI.

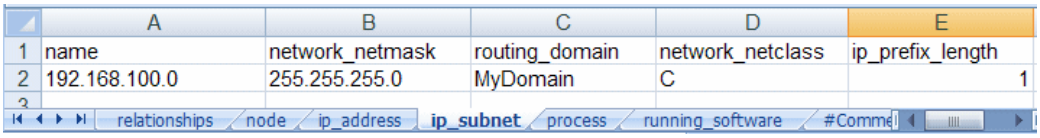
b. Use the same procedure to define IP addresses in a second Excel sheet, for example, **Sheet2**.



	A	B	C
1	ip_address	routing_domain	name
2	192.168.100.100	MyDomain	192.168.100.100
3	192.168.100.101	MyDomain	192.168.100.101
4	192.168.100.200	MyDomain	192.168.100.200
5	192.168.100.201	MyDomain	192.168.100.201

relationships | node | ip_address | ip_subnet

c. Use the same procedure to define a network CI in a third Excel sheet, for example, **Sheet3**.



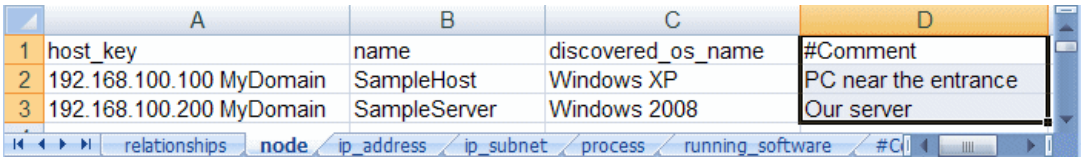
	A	B	C	D	E
1	name	network_netmask	routing_domain	network_netclass	ip_prefix_length
2	192.168.100.0	255.255.255.0	MyDomain	C	1
3					

relationships | node | ip_address | ip_subnet | process | running_software | #Comme

running_software and process definitions are described in "Add CIs with containers" below

5. **Create Comment columns - optional**

If you want to have a **Comment** column with explanations of data, use the **#** sign before the column heading. Any data placed in this column will not be imported into UCMDB.



	A	B	C	D
1	host_key	name	discovered_os_name	#Comment
2	192.168.100.100 MyDomain	SampleHost	Windows XP	PC near the entrance
3	192.168.100.200 MyDomain	SampleServer	Windows 2008	Our server

relationships | node | ip_address | ip_subnet | process | running_software | #C

6. **Add CIs with containers**

Objects that are contained within other objects cannot exist without them. For example, processes and running software cannot exist without the node they are running on. To show this relationship, a **root_container** attribute is needed. Because the container is in another CI, a reference to it is needed.

Objects can be referenced in one of the following ways:

- **By creating an Excel definition reference to the object**.

  The Excel definition referencing style is recommended because only the tab name (CI type name) and row number (the row number of the CI defined on the tab) are needed to identify any imported CI - the presence or absence of any key fields is not necessary, reconciliation rules are defined in UCMDB, and so on.

  Typical links appear as **=node!A2**, meaning that the **node** tab on the CI defined at row **2** is being referenced. It does not matter which column you are referencing; only the rows numbers are significant.

> **Note:** Such references cannot be used if the Excel file was created from a CSV file or using some other non-Excel format.

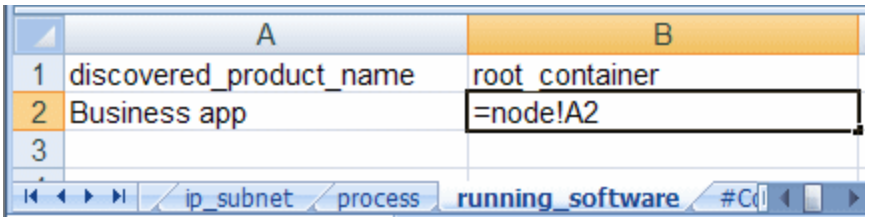
For more information about references, see Microsoft Excel documentation.

- **By setting a composition of the desired object key fields divided by the pipe symbol ('|').**

  For example, to reference an IP address, the **ip_address** and **routing_domain** attributes are needed: **192.168.100.100|MyDomain**.

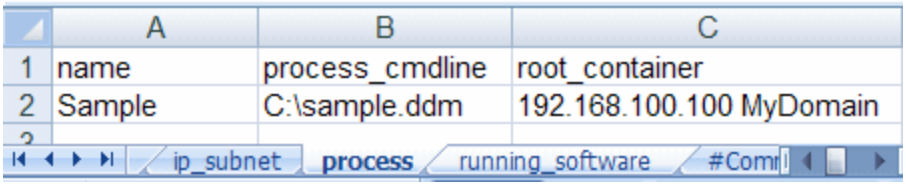
  > **Note:**
  > - The order of the key fields in the definition is important!
  > - Many objects have no keyed attributes and are identified with reconciliation rules. For this reason, Excel references are preferred.

a. Create a **running_software** using Excel references.



> **Note:** To define an Excel reference, type an equal sign (=) in a cell, select the desired reference cell, and press ENTER.

b. Create a **process** using a composite key.



7. **Define relationships**

   To define relationships, create a sheet called **relationships.**

   > **Note:** You cannot import relationship CIs.

   All links (relationships) in UCMDB are directed. This means each link has a start and end point. Also, links have names that might have some attributes similar to other CIs.

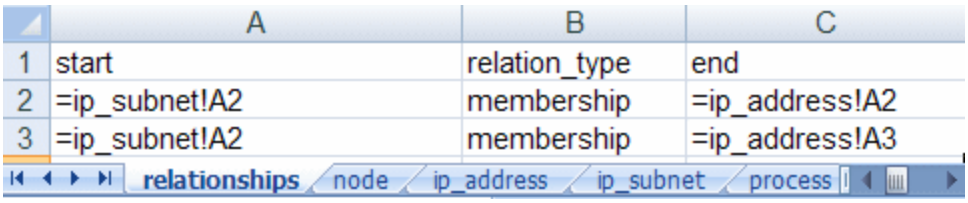
   A link definition in an import file looks as follows:

   ```
   Start object reference -> link name -> End object reference [->
   Attributes]
   ```

Link attribute definitions are described in "Add relationship attributes" on next page.

The first row (column headings) displays the reason for the information. On this sheet, only the order of the parameters is important.

a. Using Excel references, add informative captions and define member links between the IP subnet and first two IP addresses.



	A	B	C
1	start	relation_type	end
2	=ip_subnet!A2	membership	=ip_address!A2
3	=ip_subnet!A2	membership	=ip_address!A3

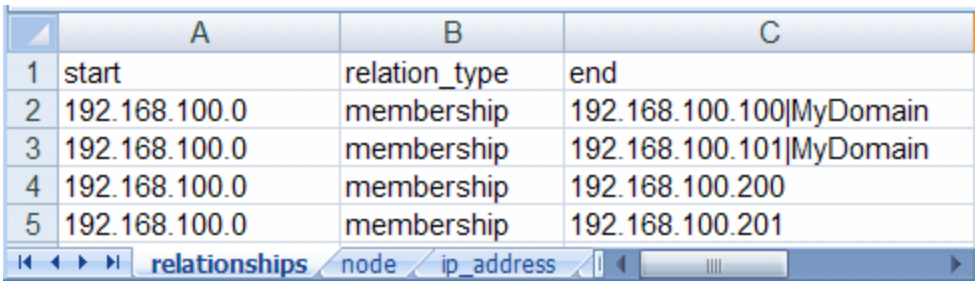
relationships | node | ip_address | ip_subnet | process

In this image, defined formulas are displayed (for example, **=ip_address!A2**). In actuality, the values of referenced cells are shown.

b. Using key composition, define the relationships between the two IP addresses and their routing domains as follows:

IP key fields are ip_address and routing_domain. The composite key looks like **192.168.100.100|MyDomain**.

The relationship tab looks as follows:



	A	B	C
1	start	relation_type	end
2	192.168.100.0	membership	192.168.100.100|MyDomain
3	192.168.100.0	membership	192.168.100.101|MyDomain
4	192.168.100.0	membership	192.168.100.200
5	192.168.100.0	membership	192.168.100.201

relationships | node | ip_address

Note:

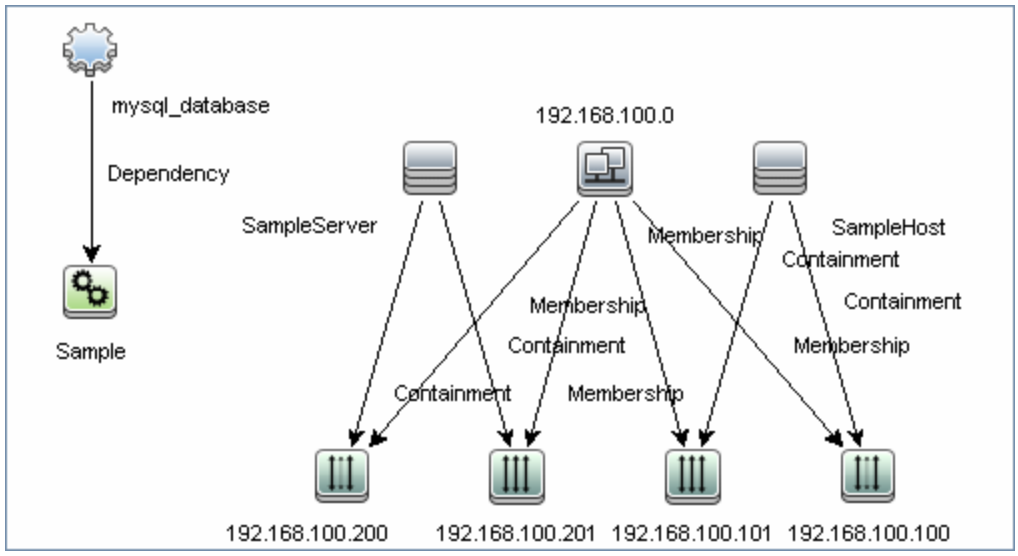
- Any type of reference can be chosen. You can use only one reference type in a cell.
- Since the IP subnet CI has no key attributes in UCMDB 9.0x, they can be referenced only by Excel reference.

c. Add a **containment** reference from **node** to **ip_address** and add a **dependency** reference from **running_software** to **process**:

	A	B	C
1	start	relation_type	end
2	192.168.100.0	membership	192.168.100.100|MyDomain
3	192.168.100.0	membership	192.168.100.101|MyDomain
4	192.168.100.0	membership	192.168.100.200
5	192.168.100.0	membership	192.168.100.201
6	192.168.100.100 MyDomain	containment	192.168.100.100|MyDomain
7	192.168.100.100 MyDomain	containment	192.168.100.101|MyDomain
8	192.168.100.200 MyDomain	containment	192.168.100.200
9	192.168.100.200 MyDomain	containment	192.168.100.201
10	Business app	dependency	C:\sample.ddm

relationships | node | ip_address | ip_s

After importing this Excel file, the topology appears as follows:



8. **Add relationship attributes**

> **Note:** This use case is not widespread, but the Import from Excel Workbook job offers such capability.

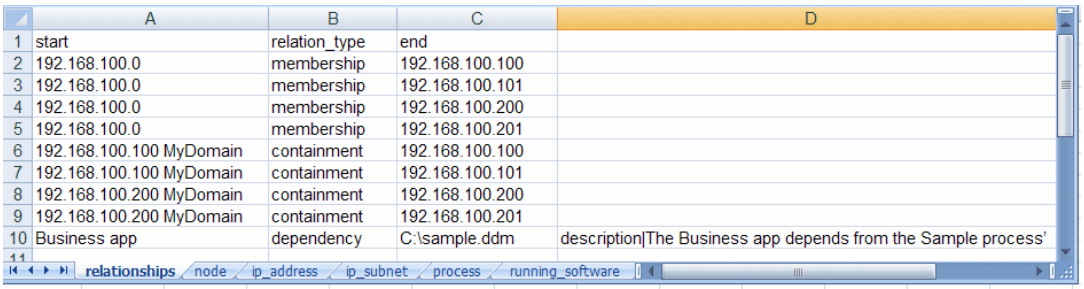
Since many different types of links can be defined on the **relationships** tab in Excel, it is impossible to name columns with attribute names. For this purpose, the following notation is used:

```
<Attribute name>< relationship_attr_delimiter><Attribute value>
```

By default, for **relationship_attr_delimiter**, a pipe symbol ('|') is used.

The description definition for the link **dependency** from running_software to process looks like **description|The Business app depends from the Sample process**.

Now the **relationships** tab appears as follows:



	A	B	C	D
1	start	relation_type	end	
2	192.168.100.0	membership	192.168.100.100	
3	192.168.100.0	membership	192.168.100.101	
4	192.168.100.0	membership	192.168.100.200	
5	192.168.100.0	membership	192.168.100.201	
6	192.168.100.100 MyDomain	containment	192.168.100.100	
7	192.168.100.100 MyDomain	containment	192.168.100.101	
8	192.168.100.200 MyDomain	containment	192.168.100.200	
9	192.168.100.200 MyDomain	containment	192.168.100.201	
10	Business app	dependency	C:\sample.ddm	description|The Business app depends from the Sample process'
11				

relationships | node | ip_address | ip_subnet | process | running_software

If many attributes must be added, they must be defined in additional columns in the **dependency** row.

> **Note:** On the **relationships** tab, no captions are needed for the attribute columns. If the column heading is present, these columns are treated as **comment** columns.

9. **Convert attribute types to UCMDB attribute types**

   At the importing stage, each attribute is converted to the type defined in the UCMDB class model. This means that if an attribute is defined in UCMDB with a text value (for example, the attribute **port** in Service Address), but in the Excel file it has an integer value (for example, **5**), it will be converted to the corresponding type.

   The following UCMDB attribute types are supported:

boolean	**date**	**double**	**enumerations**
float	**integer**	**integer_list**	**long**
string	**string_list**	**xml**	

> **Note:** If the attribute cannot be converted to the type defined in UCMDB, it is skipped and you receive a warning in the UI.

Two list types exist in UCMDBâ€"**integer_list** and **string_list**. To import such types, the value delimiters are intended. They are **integer_list_delimiter** and **string_list_delimiter** respectively. The default values are separated by a comma (**,**), but this can be changed to a job parameter.

If there is an attribute named **some_int_list** and it needs to be set using an integer list from 1 to 5, the cell in the **relationships** tab will look like:

```
some_int_list|1,2,3,4,5
```

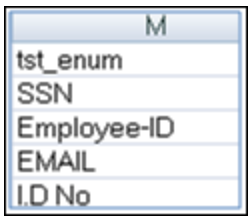
- **Enumerate attribute types**

  Enumeration data types are supported for attributes. The job assumes the enumeration has been entered in human readable form and performs a search of the internal integer representation used in UCMDB.

  If a value is entered that is not an enumeration value, it is ignored and you receive a warning in the log.

Because enumeration values are case sensitive in UCMDB, they are also case sensitive in Excel.

For example, if **SSN** in the image below had been written in lower case letters, **ssn**, the job would send an error message because it would not find the **ssn** string in UCMDB.



Import from Excel Workbook Job

Note: The Import from Excel Sample job is similar to the Import from Excel Workbook job. It differs only by reference to the sample import file.

This section includes:

Discovery Mechanism

Each tab in the Excel file reflects a specific CI type. The CIT must be defined in the UCMDB data model prior to importing file content. If only out-of-the-box CITs are imported, you do not have to create the CITs because they already exist in UCMDB.

All attributes defined for a CIT must also already exist in UCMDB or the data will be rejected. Any special rules for attributes—such as data type, obligation, formatting, and so on—must also be acceptable by UCMDB for the data to be successfully imported into UCMDB.

The data type of the attribute —string, long, integer, boolean, and so on— depends on the UCMDB data model. You do not need to set attribute types manually. You must specify the attribute name in the document header line.

Discovery performs the following validations:

1. Verifies that the CITs on the tabs in the Excel spreadsheet exist in UCMDB.
2. Verifies that the attributes (the column names in the Excel spreadsheet) exist in UCMDB.
3. Checks the presence of key attributes on the Excel spreadsheet.
4. Processes all CITs that contain a **root_container** attribute after CITs that do not have this

type of attribute. This helps to ensure that the parent CI is created before a contained CI.

5. Processes the **relationships** tab last to create relationships between CIs that do not use the containment (**container_f**) relationship.

   For the relationship to be created, the keyed attributes of a CI must be used in the relationships tab.

6. Relation attributes also must exist in the UCMDB class model.

1. Verifies that the CITs on the tabs in the Excel spreadsheet exist in UCMDB.
2. Verifies that the attributes (the column names in the Excel spreadsheet) exist in UCMDB.
3. Checks the presence of key attributes on the Excel spreadsheet.
4. Processes all CITs that contain a **root_container** attribute after CITs that do not have this type of attribute. This helps to ensure that the parent CI is created before a contained CI.
5. Processes the **relationships** tab last to create relationships between CIs that do not use the containment (**container_f**) relationship.

   For the relationship to be created, the keyed attributes of a CI must be used in the relationships tab.

6. Relation attributes also must exist in the UCMDB class model.

Trigger Query

The Import from Excel Workbook job has no trigger query. Therefore, you must manually add the Probe that imports the data. For details, see "Probe Selection Pane" in the *HP Universal CMDB Data Flow Management Guide*.

Job Parameters

Parameter	Description
file_name	The import file name. An absolute path accessible from the used probe must be used. For details on settin up this file, see " How to Set Up Import File in Excel" on page 619.
integer_list_ delimiter	The delimiter used to handle values in the spreadsheet that are to be treated as the UCMDB data type **integer_list**.
string_list_ delimiter	The delimiter used to handle values in the spreadsheet which would be mapped as the UCMDB data type **string_list**.
relationship_ attr_ delimiter	On the Relationship tab of the source file object, the linked attributes could be added. The default is **attribute_name|attribute_value** (a pipe symbol is used between the attribute name and value). This should be aligned with actual data.

Adapter

- **Input Query**

- **Input CIT:** discoveryprobemanager
- **Input query:** Because the Import from Excel Workbook job's input CIT is **Discovery Probe Gateway**, there is no need to supply an input TQL query.

- **Scripts Used**

The following scripts are used to import data from an Excel workbook.

- **import_from_excel.py**
- **xlsutils.py**

Note: The Import from Excel Workbook job may also use library scripts supplied in the Auto Discovery content package.

Created/Changed Entities

Entity Name	Entity Type	Entity Description
Import from Excel Workbook	Job	Main importing job
Import from Excel Sample	Job	Sample job that imports the predefined sample import file
XLS_Parser	Adapter	Discovery adapter
import_from_excel.py	Script	Main import script
xlsutils.py	Script	Contains utility methods for class model validation and fetching objects from Excel worksheets
ciimports_for9.xls	Resource	Sample import file
poi-3.7-beta1-20100620.jar	Resource	Java library for working with Excel 97-2003 file format
poi-ooxml-3.7-beta1-20100620.jar	Resource	Java library for working with Excel 2007 file format
poi-ooxml-schemas-3.7-beta1-20100620.jar	Resource	Java library with XML schemas used in Excel 2007 files
geronimo-stax-api_1.0_spec-1.0.jar	Resource	Geronimo implementation of standard XML processing API (used by POI)
xmlbeans-2.3.0.jar	Resource	Library for accessing XML by binding it to Java types (used by POI)

Discovered CITs

- **ConfigurationItem**

- **Managed Relationship**

Note: To view the topology, see "Topology" on page 616.

Troubleshooting and Limitations

- **Problem**: Import from Excel Workbook job compile time errors and problems working with the Excel files.

  Solution: Verify that you have performed the instructions in the Prerequisite section of the this discovery. For details, see "Prerequisite - Set up permissions" on page 617.

- **Problem**: Importing a CI with the qualifier **RANDOM_GENERATED_ID_CLASS**, but without defined reconciliation rules, leads to duplicating such CIs.

  Solution: Currently this problem is not resolvable on the job side. This can only be resolved by defining reconciliation rules.

- **Problem**: Import from Excel Workbook job date errors.

  Solution: The date cannot be imported if it is represented in text format. This issue is not resolvable because of localization. Represent the date in numerical format.

- **Limitation**: The DFM Probe breaks down the imported data into 20 KB chunks. This can cause identification issues.

Chapter 50

Importing Data from External Sources

This chapter includes:

Overview

Your data is probably stored in several formats, for example, in spreadsheets, databases, XML documents, properties files, and so on. You can import this information into HP Universal CMDB and use the functionality to model the data and work with it. External data are mapped to CIs in the CMDB.

The following external data sources are currently supported:

- "Comma Separated Value (CSV) Files" below
- "Databases" on next page
- "Properties Files" on next page

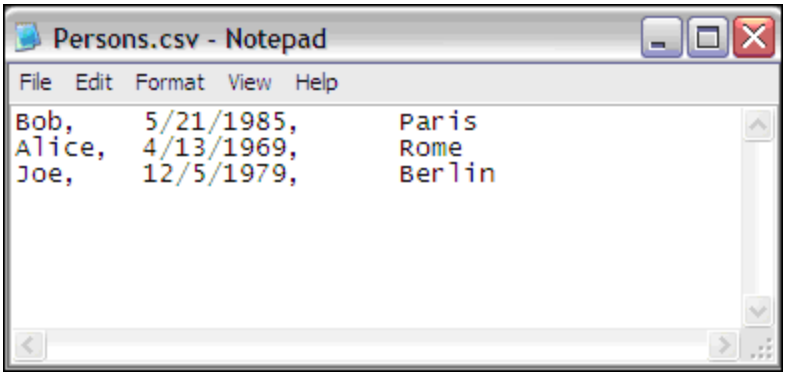
Comma Separated Value (CSV) Files

A *.csv file has a format that stores tabular data. Each row in a CSV file represents a set of values delimited with a particular delimiter. All rows are homogeneous, that is, each row has the same number of values. Values from all rows with the same index create a column. Values in a single column represent the same type of data. Therefore a CSV file represents a table of data (with rows and columns).

The default delimiter for CSV files is the comma, but any symbol can be used as a CSV delimiter, for example, a horizontal tab.

Note: Microsoft Office Excel includes native support for the CSV format: Excel spreadsheets can be saved to a CSV file and their data can then be imported into UCMDB. CSV files can be opened in an Excel spreadsheet.

Example of a CSV file:



CSV Files with Column Titles in First Row

CSV files often include column headings in the first row. When data is imported from these files, the titles are considered data and a CI is created for this row. To prevent a CI being created, you can define which row DFM should start at when importing data from a CSV file:

1. Select **Adapter Management > Resources pane > Packages > External_source_import > Adapters > Import_CSV**.
2. In the **Adapter Definition** tab, locate the **Adapter Parameters** pane.
3. Locate the **rowToStartIndex** parameter.

   By default, the value is **1**, that is, DFM retrieves data from the first row.
4. Replace **1** with the number of the row at which to start retrieving data. For example, to skip the first row and start with the second row, replace **1** with **2**.

Databases

A database is a widely used enterprise approach to storing data. Relational databases consist of tables and relations between these tables. Data is retrieved from a database by running queries against it.

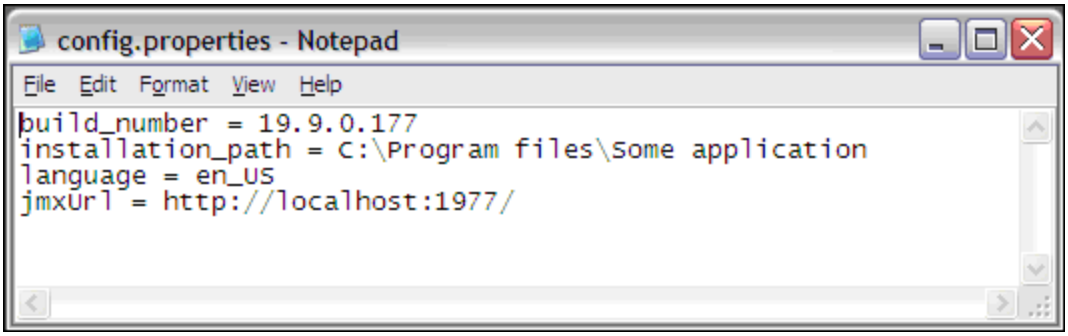
The following databases are supported: Oracle, Microsoft SQL Server, MySQL, and DB2.

Properties Files

A properties file is a file that stores data in the **key = value** format. Each row in a properties file contains one key-to-value association. In code terms, a properties file represents an associative array and each element of this array (key) is associated with a value.

A properties file is commonly used by an application to hold its configuration. If your application uses a configuration file, you can model the application in UCMDB.

Example of a properties file:



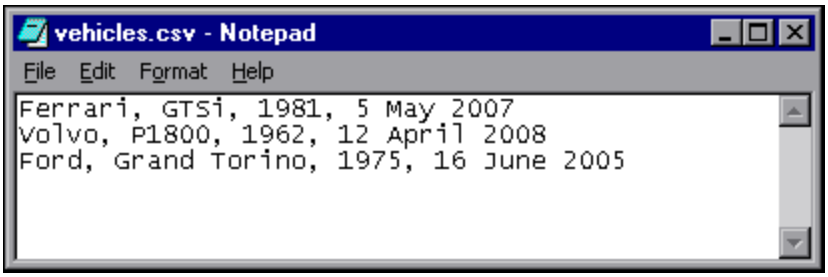
How to Import CSV Data from an External Source – Scenario

The UCMDB administrator must model a vehicle catalog that is stored in a CSV file.

This task includes the following steps:

1. ## Prerequisites

   The admin opens the CSV file and analyzes the data:

   The file includes the name, model, year of manufacture, and the date when the car was purchased, that is, there are four columns of data:

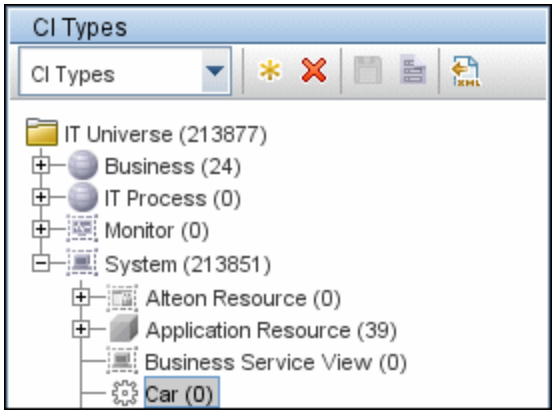
1	Name	string
2	Model	string
3	Year of manufacture	integer
4	Date of purchase	date

   There are three rows to the file, which means that the admin expects three CIs to be created in UCMDB.

2. ## Create a CIT

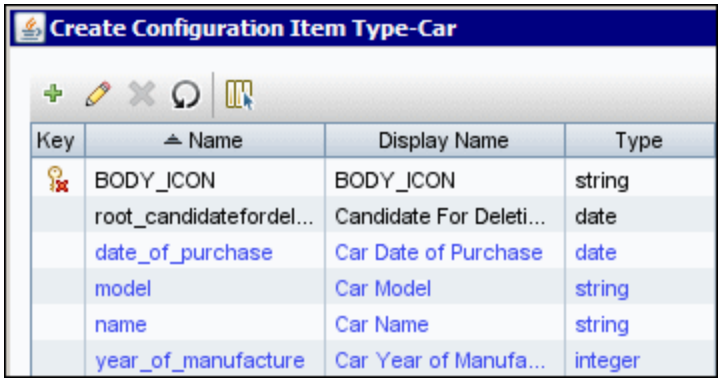
   The admin creates a CIT.

   a. The admin creates a CIT named **Car** to hold the attributes that are to be mapped to the data in the CSV file (name, model, and so on):



For details, see "Create a CI Type" in the *HP Universal CMDB Modeling Guide*.

b. During the creation of the CIT, the admin adds these attributes as follows:



For details, see "Attributes Page" in the *HP Universal CMDB Modeling Guide*.

3. Create a mapping file

The admin uses the template, **mapping_template.xml**, to create a mapping file that makes the information available to the **Import_CSV** adapter. The mapping file is located in the following folder: **Adapter Management > Resources pane > External_source_import > Configuration Files**.

a. For each attribute, the admin adds a **<map>** marker:

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\mapping_schema.xsd"
parserClassName="com.hp.ucmdb.discovery.library.
communication.downloader.cfgfiles.CiMappingConfigFile">
    <ci type="car">
        <map>
            <attribute>name</attribute>
            <column>1</column>
        </map>
        <map>
            <attribute>model</attribute>
```

```
            <column>2</column>
        </map>
        <map>
            <attribute>year_of_manufacture</attribute>
            <column>3</column>
        </map>
        <map>
            <attribute>date_of_purchase</attribute>
            <column>4</column>
        </map>
    </ci>
</mappings>
```

b. The admin then adds information about the attribute type:

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\mapping_schema.xsd"
parserClassName="com.hp.ucmdb.discovery.library.
communication.downloader.cfgfiles.CiMappingConfigFile">
    <ci type="">
    <map>
        <attribute>name</attribute>
        <column>1</column>
    </map>
    <map>
        <attribute>model</attribute>
        <column>2</column>
    </map>
    <map>
        <attribute>year_of_manufacture</attribute>
        <column>3</column>
        <converter module="import_
converters">stringToInteger</converter>
    </map>
    <map>
        <attribute>date_of_purchase</attribute>
        <column>4</column>
        <converter module="import_
converters">stringToDate</converter>
    </map>
</mappings>
```

All conversions between the values in the CSV file and the CI attributes are done by a converter. Several converter types are included in the package by default. For details, see "How to Convert Strings to Numbers" on page 639.

4. Run the job

How you run this job depends on the software version being used.

UCMDB 9.04 (and later), with CP10

In DFM, in the Integration Studio, create a new integration point.

- Provide a name and description for the integration point.
- Under **Integration Properties > Adapter**, select the **Import from CSV** adapter.
- Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.
- Under **Adapter Properties > Trigger CI instance** select:
  - **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI; or
  - **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

  Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.
- Save the Integration Point.
- Run the job.

Note: For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

UCMDB 9.03 (and earlier)

This job uses the **Shell Trigger** CIT to discover the CSV file on a remote machine. The Input CIT is **Shell** and the discovered CIT is declared as **IT Universe**. However, the actual discovered CIs depend on the mapping configuration.

The admin activates the following job: **Import from CSV file**.

For details on activating jobs, see "Discovery Modules Pane" in the *HP Universal CMDB Data Flow Management Guide*.

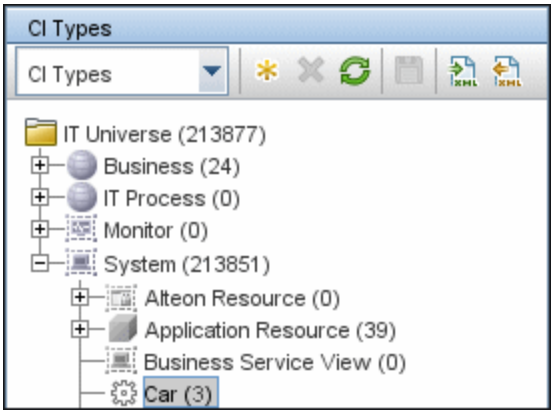
5. ## Add the discovered Shell CI to the job

Note: This step only applies if using **UCMDB 9.03** and earlier.

After activation, the admin locates the `Shell` CI (of the machine where the `cars.csv` file is located) and adds it to the job. For details, see "Choose CIs to Add Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

6. ## Result

The admin accesses the CIT Manager and searches for instances of the **Car** CIT. UCMDB finds the three instances of the CIT:


CI Types
CI Types
IT Universe (213877)
Business (24)
IT Process (0)
Monitor (0)
System (213851)
Alteon Resource (0)
Application Resource (39)
Business Service View (0)
Car (3)

How to Convert Strings to Numbers

Converters enable you to specify the way data should be converted between the external source and a CI's attributes.

A CSV file contains records of type `string`. However, some of the record values need to be handled as numbers. This is done by adding a **converter** element to the **map** element (in **`[your mapping file name].xml`**):

```
<converter module="import_converters"></converter>
```

The **import_converters.py** file (**Adapter Management > Resources pane > Packages> External_source_import > Scripts)** contains a set of the most commonly needed converters and types:

- toString
- stringToInt
- stringToLong
- stringToFloat
- stringToBoolean
- stringToDate
- stringToDouble
- skipSpaces
- binaryIntToBoolean
- stringToBytesArray
- stringToZippedBytesArray

Example of a Converter

A CSV file contains the following row:

```
Usain, 21, Male
```

This row must be mapped to the **Person** CIT that includes name (`Usain`), age (`21`), and gender (`Male`) attributes. The `age` attribute should be of type **integer**. Therefore, the string in the CSV file must be converted to an integer in the CIT to make it compliant with the CIT attribute type, before the Person CIs can retrieve the `age` values.

This is done by adding a **converter** element to the **map** element:

```
<map>
    <attribute>age</attribute>
    <column>2</column>
    <converter module="import_converters">stringToInt</converter>
</map>
```

module="import_converters". This attribute specifies from which module the converter is to be retrieved. A module is a Jython script file that contains a set of converter methods, in this case, `import_converters.py`.

stringToInt. The name of the converter. In the `import_converters.py` file, the method is written as follows:

```
def stringToInt(value):
    if value is not None:
        return int(value.strip())
    else:
        return 0
```

Custom Converters

You can write your own custom converters: Add a new method to the `import_converters.py` file or create your own script and add a set of converter methods to it. Call the method with the name of the script, for example:

```
<converter module="your_converter_script">[your_converter_method]
</converter>
```

External_source_import Package

The **External_source_import** package consists of three jobs and three adapters. There is one job and one adapter for each external source (CSV file, properties file, database):

External Source	Job	Adapter
CSV file	Import from CSV file	Import_CSV
Properties file	Import from Properties file	Import_Properties_file
Database	Import from Database	Import_DB

The jobs are located under **Discovery Control Panel > Discovery Modules/Jobs > Discovery Modules > Others > Discovery Tools**.

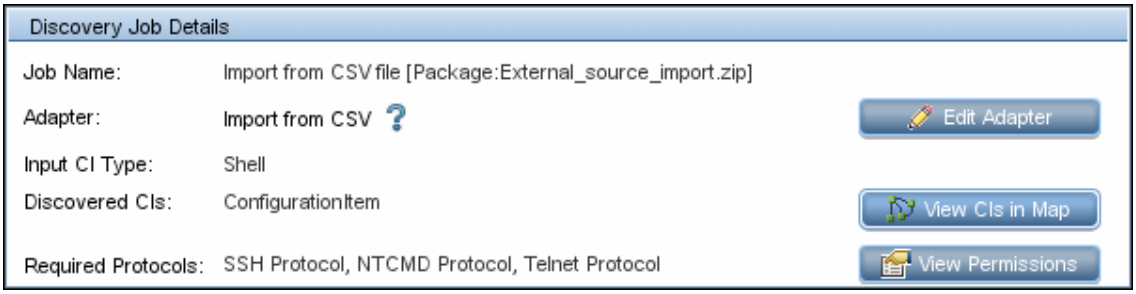
The adapters are located in the **Integration Studio** and are accessed when creating or editing an Integration Point.

Import from CSV File Job

This section includes the following topics:

Job Details

The job details are as follows:



This job has no Trigger queries associated with it. That is, this job is not triggered automatically (nor are the `Import from Properties file` and the `Import from Database` jobs). After you activate the job, you must manually add input CIs to the job so that it runs against a particular destination. For details, see "Add the discovered Shell CI to the job" on page 637.

Adapter Parameters

The following parameters are included by default:

Parameter	Description
bulkSize	This parameter only works if the parameter **flushObjects** is **true**, in which case, when sending discovery results, it sets the size of chunks used to that number of CIs. The **default** is 2,000 CIs.
ciType	The CIT name. This job creates and reports CIs of this type to UCMDB, based on data in the CSV file. For example, if the CSV file contains records for UNIX hosts, you must set the **ciType** parameter to **unix**.
csvFile	The full path to the CSV file on the remote machine. The job uses the Shell CI Type as input to reach this path on the remote machine.
delimiter	The delimiter used in the CSV file. The comma (,) delimiter is the default but other delimiters are supported. For details, see "Delimiters" on page 643.

Parameter	Description
flushObjects	This parameter allows customization of the reporting mechanism. If **true**, the probe divides the discovery result into chunks, and sends each chunk to the UCMDB Server. This helps prevent out-of-memory issues where a large amount of data is sent. The chunk size can be configured with the **bulkSize** parameter. If **false** (the default value), the probe sends the discovery result without dividing it into chunks.
mappingFile	For details of the mapping file, see "External Source Mapping Files" on page 650.
mappingString	The string containing mapping information used to map the CSV column indexes and attributes to import. You define this mapping in the following format: • mapping elements should be separated by commas • each mapping element should be specified in a <column number>:<attribute name> format, for example: The string 0:host_key,1:name defines the mapping of two attributes of a host CI, where the host's host_key attribute is taken from the value in the first column (0) and the name attribute is taken from the value in the second column (1)
quoteSymbol	Quoting symbol used in the CSV file. **Default symbol:** "
rowToStartIndex	For details on setting the row at which DFM starts collecting data, see "CSV Files with Column Titles in First Row" on page 633.

For details on overriding an adapter parameter, see "Override Adapter Parameters" in the *HP Universal CMDB Developer Reference Guide*.

Mapping Information for the Import from CSV File Job

You can specify mapping information for the **Import from CSV File** job with one of the following methods:

- In an external XML file. You must specify the **mappingFile** parameter. For details, see "External Source Mapping Files" on page 650.
- Directly in a job's **ciType** and **mappingString** parameters, without using an external file.

> **Note:** When using this mapping method, you cannot specify attribute types or converters.

If the **mappingFile** parameter is specified, the job tries to retrieve mapping information from the XML file. If it is not specified, the job uses the mapping information specified in the **ciType** and **mappingString** parameters.

Delimiters, Quotes, and Escaping Characters

- **Delimiters**

  The delimiter divides values in the same row of a CSV file. Supported delimiters are:

  - **Single symbol.** Any symbol can be used as a delimiter, for example, the pipe sign (**|**), the letter **O**. Delimiters are case sensitive.
  - **ASCII code.** If an integer number is used as the value for a delimiter parameter, this value is treated as ASCII code, and the related symbol is used as the delimiter. For example, **9** is a valid delimiter because **9** is the ASCII code for the horizontal tab.
  - **Known character sequence**. A sequence of characters can be used to represent special characters. For example, **\t** represents the horizontal tab.

- **Quotation Marks**

  You can use double or single quotes in values, that is, all values residing between the two quotes are treated as a single value.

  - If a delimiter symbol is used in a value, the value must be surrounded with quotation marks. For example, the following row includes a comma inside a value, so the value must be quoted:

    ```
    Morganfield, "25 Hope Road, Kingston", Jamaica
    ```

  - If a quote character is used in a value, the character must be escaped by inserting a backslash before it:

    ```
    McKinley \"Muddy Waters\" Morganfield, "April 4, 1915"
    ```

    This row contains two values:

    - McKinley "Muddy Waters" Morganfield

    - April 4, 1915.

- **Escaping Symbols**

  The following symbols must always be quoted or escaped:

  - Backslash
  - Single quote
  - Double quote
  - Delimiter, that is, the delimiter used in the same CSV file.

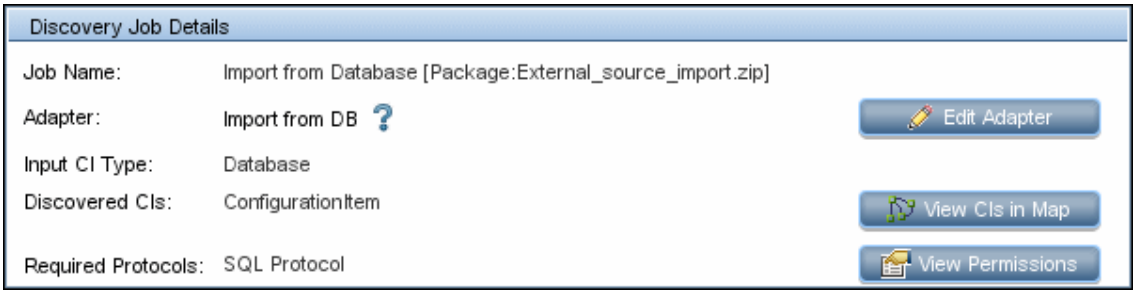
Import from Database Job

This job uses a database table or database query as the source of the information, maps the information to CIs, and imports the CIs into UCMDB.

This section includes the following topics:

Job Details

The job details are as follows:



This job has no trigger queries associated with it. The job tries to get the Instance name and Port using the attributes **Name** and **Application Listening Port Number** of the **Input Database** CI. If these attributes are empty, it uses the Instance Name and Port number defined in Generic DB Protocol (SQL) credentials.

Discovery Adapter Parameters

The following parameters are included by default:

Parameter	Description
bulkSize	This parameter only works if the parameter **flushObjects** is **true**, in which case, when sending discovery results, it sets the size of chunks used to that number of CIs. The **default** is 2,000 CIs.
ciType	Name of CIT to import.

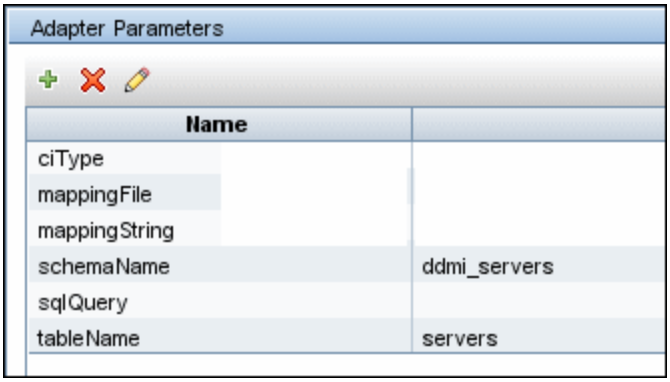
Parameter	Description
flushObjects	This parameter allows customization of the reporting mechanism. If **true**, the probe divides the discovery result into chunks, and sends each chunk to the UCMDB Server. This helps prevent out-of-memory issues where a large amount of data is sent. The chunk size can be configured with the **bulkSize** parameter. If **false** (the default value), the probe sends the discovery result without dividing it into chunks.
mappingFile	XML file containing the mapping from column to attribute.
mappingString	The string containing mapping information used to map the Database column names and the attributes to import. You define this mapping in the following format: • mapping elements should be separated by commas; • each mapping element should be specified in a <column name>:<attribute name> format, **Example:** A_IP_ADDRESS:ip_address, A_IP_DOMAIN:ip_domain
schemaName	The name of the database schema.
sqlQuery	If a SQL query is specified, mapping is performed against its result. This parameter is ignored if **tableName** is defined.
tableName	If a table name is specified, mapping is performed against the table's columns.

For details on overriding an adapter parameter, see "Override Adapter Parameters" in *HP Universal CMDB Developer Reference Guide*.

Tables and Queries

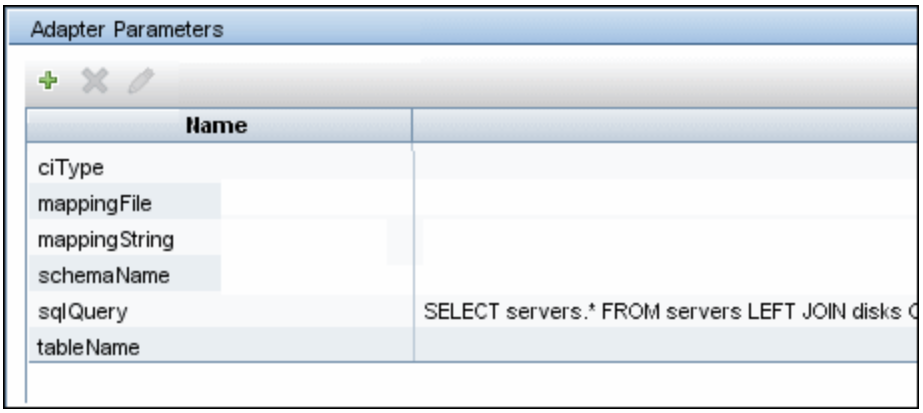
The following use cases are supported by the `Import from Database` job (a single SQL query is performed):

- Import data using the schema name and table name parameters:



The SQL query is generated from these parameters.

- Import data specifying an arbitrary SQL query as the source of the data:



The SQL query is generated from the defined query. For more details, see "Importing Data with an SQL Query" below.

Database, Schema, and Table Names

SQL naming conventions suggest a usage of a `<database.schema.table>` syntax for the fully qualified name of a table. Note, however, that each vendor treats the specification in a different way. DFM uses the following notation:

- The **schemaName** parameter specifies the name of a database.
- The **tableName** parameter specifies the name of a table.
- A schema name cannot be specified in a parameter but can be included in a SQL query.

For Oracle, the SQL query is:

```
SELECT * FROM <schemaName.tableName>
```

For Microsoft SQL Server, the SQL query is:

```
SELECT * FROM dbo.tableName
```

Note: The default `dbo` schema is used for Microsoft SQL Server.

Importing Data with an SQL Query

You can use arbitrarily-complex SQL query expressions, for example, joins, sub-selects and other options, as long as the query is valid and complies with the database usage. Currently, you must use a fully-qualified table name in the query according to the specific database.

Column Types

Types enable you to specify, in the mapping file, the type of column that exists in the external source. For example, a database includes information about column types, and the value of this type needs to be included in the CI's attributes. This is done by adding a **type** element to the **map** element (in **`mapping_[your mapping file name].xml`**):

```
<column type="int"></column>
```

Supported type attributes are:

- string
- Boolean
- date
- int
- long
- double
- float
- timestamp

Note:

- You use the **type** attribute for database mapping only.
- If the column element does not include a `type` attribute, the element is mapped as a string.

Example of adding a type attribute

A database column has an integer type and can be either 0 or 1. This integer must be mapped to a Boolean attribute of a CIT in UCMDB. Use the binaryIntToBoolean converter, as follows:

```
        <map>
            <attribute>cluster_is_active</attribute>
            <column type="int">cluster_is_active</column>
            <converter module="import_
converters">binaryIntToBoolean</converter>
        </map>
```

type="int". This attribute specifies that the value of `cluster_is_active` should be retrieved as an integer, and that the value passed to the converter method should be an integer.

If the `cluster_is_active` attribute of the CIT is of type `integer`, the converter is not needed here, and the mapping file should say:

```
        <map>
            <attribute>cluster_is_active</attribute>
            <column type="int">cluster_is_active</column>
        </map>
```

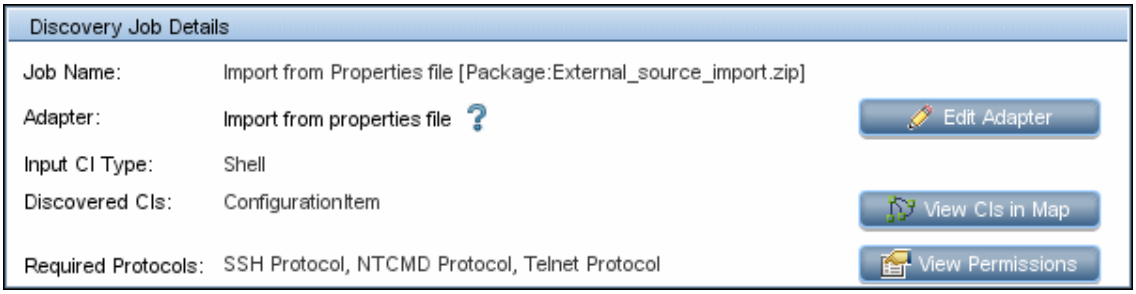
Import from Properties File Job

This job imports information from a properties file, maps the information to one CI, and imports that CI into UCMDB.

This section includes the following topics:

Job Details

The job details are as follows:



This job has no Trigger queries associated with it.

Discovery Adapter Parameters

The following parameters are included by default:

Parameter	Description
bulkSize	This parameter only works if the parameter **flushObjects** is **true**, in which case, when sending discovery results, it sets the size of chunks used to that number of CIs. The **default** is 2,000 CIs.
ciType	The name of the CIT to import.
flushObjects	This parameter allows customization of the reporting mechanism. If **true**, the probe divides the discovery result into chunks, and sends each chunk to the UCMDB Server. This helps prevent out-of-memory issues where a large amount of data is sent. The chunk size can be configured with the **bulkSize** parameter. If **false** (the default value), the probe sends the discovery result without dividing it into chunks.
mappingFile	For details of the mapping file, see "External Source Mapping Files" on page 650.

Parameter	Description
mappingString	The string containing mapping information used to map the column indexes and attributes to import. You define this mapping in the following format: • mapping elements should be separated by commas • each mapping element should be specified in a <column number>:<attribute name> format, for example: The string 0:host_key,1:name defines the mapping of two attributes of a host CI, where the host's host_key attribute is taken from the value in the first column (0) and the name attribute is taken from the value in the second column (1).
propertyFile	The full path to the properties file located on a remote machine. The Input CI runs the Shell discovery that is used to access this file on the remote machine

For details on overriding an adapter parameter, see "Override Adapter Parameters" in the *HP Universal CMDB Developer Reference Guide*.

Keys and Values

Keys cannot contain the equals symbol (**=**).

Each value must be set out in a single line. Use **backslash+n** (**\n**) to specify a new line. Values can contain anything, including **\n** for a new line, quotes, tabs, and so on.

Comments in Properties Files

To create a commented line in a properties file, add the pound sign (**#**) as the first character in a line. The job ignores commented lines.

External Source Mapping Files

The data in the external source is mapped to a CI's attributes in UCMDB by means of a mapping file. The mapping files are located in the **Adapter Management > Resources pane > Packages > External_source_import > Configuration Files** folder:

- **mapping_template.xml**. A template that serves as a source for creating the mapping file.
- **mapping_schema.xsd**. The XML schema used to validate the XML mapping file. The XML mapping file must be compliant with this schema.
- **mapping_doc.xml**. A file that contains Help on creating a mapping file, including all valid elements.

The mapping file describes the mapping only and does not include information about how data should be obtained. In this way, you can use one mapping file across different jobs.

All the adapter files in the **External_source_import** package include a `mappingFile` parameter, for example:

```
<parameter name="mappingFile" type="string" description="Mapping file
located in &quot;Configuration Files&quot; folder of this package" />
```

name="mappingFile". The value of this parameter is the mapping XML file. The mapping file is always located on the server and is downloaded to the Data Flow Probe machine upon job execution.

Troubleshooting and Limitations

- **Problem**: When CIs imported from a CSV file are displayed in the Statistics Results pane, one more CI than expected is included in the results. This is because the first row of the CSV file contains column headings that are considered as CIs.

  Solution: For details on defining from which row DFM should read the CSV file, see "CSV Files with Column Titles in First Row" on page 633.

- **Problem**: When importing large CSV or properties files on the network, there may be time-out issues.

  Solution: Make sure the files are not large.

- **Limitation**: When importing data from an external database, and the data includes a null value, it is sent to UCMDB with an attribute value of None.

- **Limitation**: The DFM Probe breaks down the imported data into 20 KB chunks. This can cause identification issues.

Chapter 51

Microsoft SCCM/SMS Integration

This chapter includes:

Overview

This document includes the main concepts, tasks, and reference information for integration of Microsoft System Center Configuration Manager (SCCM)/Systems Management Server (SMS) with HP Universal CMDB.

Integration occurs by populating the UCMDB database with devices, topology, and hierarchy from SCCM/SMS and by federation with SCCM/SMS supported classes and attributes.

According to UCMDB reconciliation rules, if a CI (in SCCM/SMS) is already mapped to a CI in the CMDB, it is updated; otherwise, it is added to the CMDB.

Microsoft System Center Configuration Manager/Systems Management Server are used by IT administrators to manage client computers and servers.

SCCM/SMS enable you to:

- manage computers that roam from one location to another
- track deployment and use of software assets, and use this information to plan software procurement and licensing
- provide IT administrators and management with access to data accumulated by SCCM/SMS
- provide scalable hardware and software management
- manage security on computers running Windows operating systems, with a minimal level of administrative overhead

Supported Versions

Integration has been developed and tested on HP Universal CMDB version 8.03 or later, with SCCM version 2007 or SMS version 2003.

SMS Adapter

Integration with SCCM/SMS is performed using an SMS adapter, which is based on the Generic DB Adapter. This adapter supports full and differential population for defined CI types as well as federation for other CI types or attributes.

The SMS Adapter supports the following features:

- Full replicating of all instances of the selected CI types.
- Identifying changes that have occurred in SCCM/SMS, to update them in the UCMDB.
- Simulating the touch mechanism capabilities:

  When a CI is removed from SCCM/SMS, it is physically deleted from the database and there is no way to report about it. The SMS Adapter supports a full synchronization interval. This means that the adapter transfers data for which the aging mechanism has been enabled, and provides the time interval to run a full synchronization that simulates the touch mechanism.
- Federation of selected CI types and attributes.

Out-of-the-box integration with SCCM/SMS includes population of the following classes:

- Node (some of the attributes are populated and some are federated)
- Layer2 connection
- Location that is connected to the node
- IP address
- Interface

In addition, the following classes can be defined as federated from SCCM/SMS:

- CPU
- File system
- Installed software
- Windows service

The following classes and attributes should be marked as federated by the SCCM/SMS adapter for the proper functionality of the Actual State feature of Service Manager:

- Classes
  - CPU
  - Installed software
  - Windows service
- Node attributes

- DiscoveredOsVendor
- DiscoveredModel
- Description
- DomainName
- NetBiosName

> **Note:** Avoid marking the **LastModifiedTime** attribute as federated, as it may lead to unexpected results.

How to Populate the CMDB with Data from SCCM/SMS

This task describes how to install and use the SMS adapter.

This task includes the following steps:

- "Define the SMS integration" below
- "Define a population job (optional)" on next page
- "Run the population job" on next page

1. **Define the SMS integration**

   a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.

   b. Click the **new integration point** button to open the new integration point Dialog Box.

   - Click , select the Microsoft SMS adapter, and click **OK**.

     Each out-of-the-box adapter comes predefined with the basic setup needed to perform integration with UCMDB. For information about changing these settings, see "Integration Studio Page" in the *HP Universal CMDB Data Flow Management Guide*.

   - Enter the following information, and click **OK**:

Name	Description
Credentials	Allows you to set credentials for integration points. For credential information, see "Supported Protocols" on page 82.
Hostname/IP	The host name of the machine where the database of SCCM/SMS is running.
Integration Name	The name you assign to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to

Name	Description
	set up an integration point without actually connecting to a remote machine.
Port	The port through which you access the MSSQL database.

c. Click **Test connection** to verify the connectivity, and click **OK**.

d. Click **Next** and verify that the following message is displayed: **A connection has been successfully created**. If it does not, check the integration point parameters and try again.

2. ## Define a population job (optional)

The Microsoft SMS adapter comes out-of-the-box with the **hostFromSMS Population** job, which runs the following predefined query: **hostDataFromSMS**. For details about this query, see "Predefined Query for Population Jobs" on page 659. This job runs according to a default schedule setting.

You can also create additional jobs. To do this, select the Population tab to define a population job that uses the integration point you defined in "Define the SMS integration" on previous page. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

3. ## Run the population job

Activate the population job in one of the following ways:

- To immediately run a full population job, click  . In a full population job, all appropriate data is transferred, without taking the last run of the population job into consideration.
- To immediately run a differential population job, click  . In a differential population job, the previous population time stamp is sent to SCCM/SMS, and SCCM/SMS returns changes from that time stamp to the present. These changes are then entered into the UCMDB database.
- To schedule a differential population job to run at a later time or periodically, define a scheduled task. For details, see "Define Tasks that Are Activated on a Periodic Basis" in the *HP Universal CMDB Administration Guide*.

Note: the replicated CIs are controlled by the integration TQL that is used. You can create additional TQL queries that contain different topologies for use in other jobs.

How to Federate Data with SCCM/SMS

The following steps describe how to define the CI types that will be federated with SCCM/SMS.

1. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
2. Select the integration point that you defined in "Define the SMS integration" on page 655.
3. Click the **Federation** tab. The panel shows the CI types that are supported by the SMS adapter.
4. Select the CI types and attributes that you want to federate.
5. Click **Save**.

Note:

- CI types that populate UCMDB should not be selected for federation. Specifically, avoid federating node, IP address, interface, location, and Layer2, which populate UCMDB out-of-the-box.
- Other CI types can be used in federation only after the node data has been replicated to CMDB by the hostDataImport query. This is because the default reconciliation rule is based on node identification.

How to Customize the Integration Data Model in UCMDB

Out-of-the-box CIs for SCCM/SMS integration can be extended in one of the following ways:

To add an attribute to an existing CI type:

If the attribute you want to add does not already exist in the CMDB, you need to add it. For details, see "Add/Edit Attribute Dialog Box" in the *HP Universal CMDB Modeling Guide.*

1. Navigate to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > SMS Adapter > Configuration Files > orm.xml**.
2. Locate the **generic_db_adapter.[CI type]** to be changed, and add the new attribute.
3. Ensure that the TQL queries that include this CI type have the new attribute in their layouts as follows:
   a. In the Modeling Studio, right-click the node where you want to include the attribute.
   b. Select **Query Node Properties**.
   c. Click **Advanced Layout Settings** and select the new attribute.

   For details about selecting attributes, see "Layout Settings Dialog Box" in the *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see "Troubleshooting and Limitations" on page 663.

To add a new CI Type to the Generic DB Adapter:

1. In UCMDB, create the CI Type that you want to add to the adapter, if it does not already exist. For details, see "Create a CI Type" in the *HP Universal CMDB Modeling Guide*.
2. Navigate to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > SMS Adapter > Configuration Files > orm.xml**.
3. Map the new CI type by adding a new entity called **generic_db_adapter.[CI type]**.

   For more details, see "The orm.xml File" in the *HP Universal CMDB Developer Reference Guide*.
4. Create queries to support the new CI types that you have added. Make sure that all mapped attributes are selected in the Advanced Layout settings:
   a. In the Modeling Studio, right-click the node where you want to include the attribute.
   b. Select **Query Node Properties**.
   c. Click **Advanced layout settings** and select the new attribute.

   For details about selecting attributes, see "Layout Settings Dialog Box" in *HP Universal CMDB Modeling Guide.* For limitations on creating this TQL query, see "Troubleshooting and Limitations" on page 663.
5. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
6. Edit the SMS integration point to support the new CI type by selecting it either for population or

for federation.

7. If the new CI type is for population, edit the population job that you created above.

Predefined Query for Population Jobs

The following TQL query is provided out-of-the-box if you use the Microsoft SMS adapter when you create an integration point:

- **hostDataFromSMS**. Imports nodes and their related data. Information also includes each node's IP address and interface.

SCCM/SMS Integration Package

This section includes:

- "Transformations" below
- "SCCM/SMS Plug-in " on next page
- "Reconciliation" on page 661

Transformations

Following is the list of transformations that are applied to values when they are transferred to or from the SCCM/SMS database:

CMDB Class	Attribute	Transformation
windows	nt_servicepack	Represents number of the Windows service pack. SCCM/SMS DB: Service Pack 2 UCMDB: 2.0 Transformer: standard GenericEnumTransformer, mapped in the **nt.nt_servicepack.transformer.xml** file.
node	host_isdesktop	A Boolean value that determines whether a machine is a desktop or a server. SCCM/SMS DB: Workstation or Server UCMDB: true or false Transformer: standard GenericEnumTransformer, mapped in the **node.host_isdesktop.transformer.xml** file.
node	host_os	Represents the node's operation system. **SCCM/SMS DB**. Microsoft Windows XP Professional **UCMDB**. Windows XP

CMDB Class	Attribute	Transformation
		Transformer. Standard **GenericEnumTransformer**, mapped in the **node.discovered_os_ name.transformer.xml** file. If the SCCM/SMS operation system value is not listed in the **transformer.xml** file, the original value is sent to UCMDB. By default, only Windows operating systems are mapped.
node	host_osinstalltype	Represents the Windows OS edition. **SCCM/SMS DB**. Microsoft Windows XP Professional **UCMDB**. Professional **Transformer**. Standard **GenericEnumTransformer**, mapped in the **host.host_ osinstalltype.transformer.xml** file. **Note**: The same column in the SCCM/SMS database is mapped to two different UCMDB attributes, using different transformers.
disk device	name	Represents the partition name. **SCCM/SMS DB**. C: **UCMDB**. C **Transformer**. standard **AdapterToCmdbRemoveSuffixTransformer** that removes the colon.
interface	interface_macaddr	Represents the MAC address of NIC. **SCCM/SMS DB**. AB:CD:EF:01:23:45 **UCMDB**. ABCDEF012345 **Transformer**. custom **SmsMacAddressTransformer** that removes the colons from the SCCM/SMS MAC address while making it compatible with the UCMDB MAC addresses.

SCCM/SMS Plug-in

The **SmsReplicationPlugin** provides enhanced functions to those found in the Generic Database Adapter. It is called when:

- full topology is requested (**getFullTopology**) – this returns all the CIs that were found in the external SCCM/SMS database.

- topology layout is requested (**getLayout**)
- topology of changes is requested (**getChangesTopology**) – this returns only the CIs that are modified or added after a specific time. The topology of the changes is calculated as follows:
  - There is a specific date (**fromDate**) after which all changes are requested.
  - Most of the entities in the SCCM/SMS database contain a Timestamp column that contains the date and time of the last modification. This Timestamp column is mapped to the **root_updatetime** attribute of a CI. Currently, some entities do not contain any creation time information. The entities that have a timestamp column must be listed in the **replication_config.txt** file.
  - In the integration TQL query, the node CI is named **Root**.
  - Using the plug-in, the integration TQL query is dynamically modified so that each **Root** entity and all entities that are listed in the **replication_config.txt** file have an additional condition causing the value of the **root_updatetime** attribute to be greater than or equal to the **fromDate** value.
  - This modified TQL query is then used to obtain the data.

Reconciliation

The adapter uses the default reconciliation rule-based mapping engine.

SMS Adapter Configuration Files

The adapter includes the following configuration files:

- **orm.xml**. The Object Relational mapping file, which maps between SCCM/SMS database tables and columns, and UCMDB classes and attributes. Both CIs and links are mapped.
- **fixed_values.txt**. Used by the Generic DB Adapter to set the **ip_domain** of IP Address CIs to `DefaultDomain`.
- **plugins.txt**. Contains configuration information for the Generic DB Adapter. Also defines three plug-ins that are used during replication: getFullTopology, getChangesTopology, and getLayout.
- **transformations.txt**. Contains the configuration for transformation of attribute values. For a list of the transformations, see "Transformations" on page 659.
- **node.discovered_os_name.transformer.xml**. Mapping used by the transformer for the **host_isdesktop** attribute.
- **node.host_osinstalltype.transformer.xml**. Mapping used by the transformer for the **host_os** attribute.
- **host.host_osinstalltype.transformer.xml**. Mapping used by the transformer for the **host_osinstalltype** attribute.
- **nt.nt_servicepack.transformer.xml**. Mapping used by the transformer for the **nt_servicepack** attribute.
- **replication_config.txt.** Contains a comma-separated list of non-root CIs and relations types that have a **timestamp** condition in the SCCM/SMS database. This status condition indicates

- **reconciliation_types.txt**. Defines the CI types that are used for reconciliation.

For details on adapter configuration, see "Developing Generic Database Adapters" in the *HP Universal CMDB Developer Reference Guide*.

Troubleshooting and Limitations

- Queries that are used in population jobs should contain one CI type that is labeled with a Root prefix, or one or more relations that are labeled with a Root prefix.

  The root node is the main CI that is synchronized; the other nodes are the contained CIs of the main CI. For example, when synchronizing the Node CI Type, that graph node is labeled as Root and the resources are not labeled Root.

- The TQL graph must not have cycles.
- A query that is used to synchronize relations should have the cardinality 1...* and an OR condition between the relations.
- The adapter does not support compound relations.
- Entities that are added in SCCM/SMS are sent as updates to UCMDB by the SMS Adapter during differential population.
- ID conditions on the integration TQL query are not supported.
- The TQL graph should contain only CI types and relations that are supported by the SCCM/SMS adapter.

Chapter 52

NetApp SANscreen/OnCommand Insight Integration

This chapter includes:

Overview

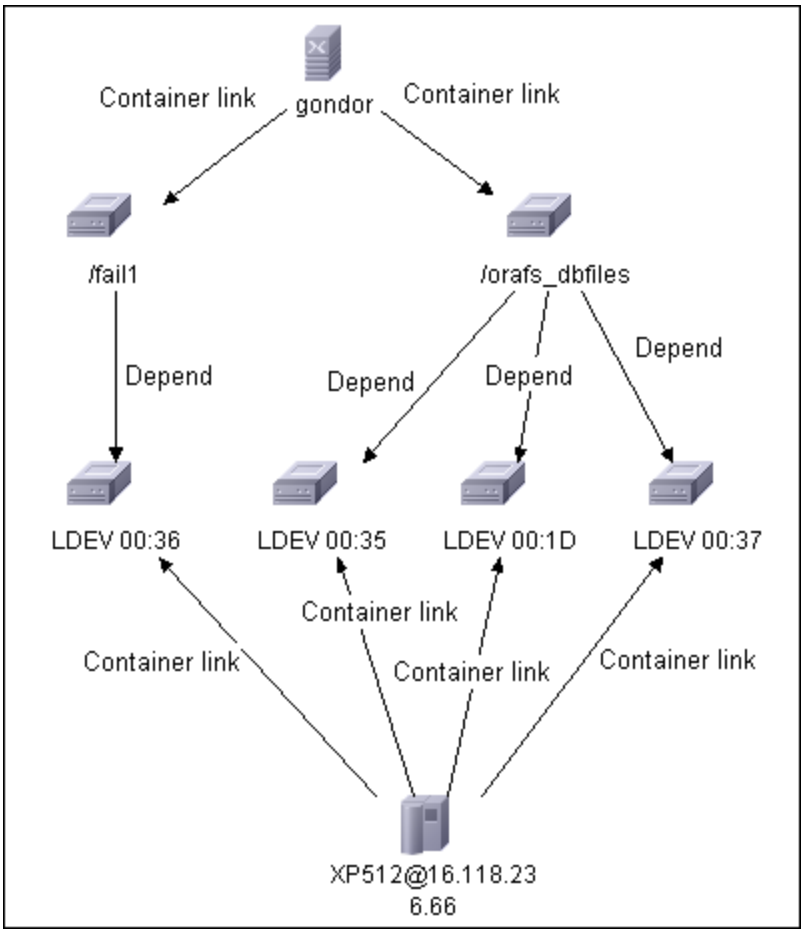
Integration between NetApp SANscreen and DFM involves a UCMDB initiated integration adapter on the SANscreen WebService API, and synchronizes devices, topology, and hierarchy of storage infrastructure in UCMDB. This enables Change Management and Impact Analysis across all business services mapped in UCMDB from a Storage point of view.

Supported Versions

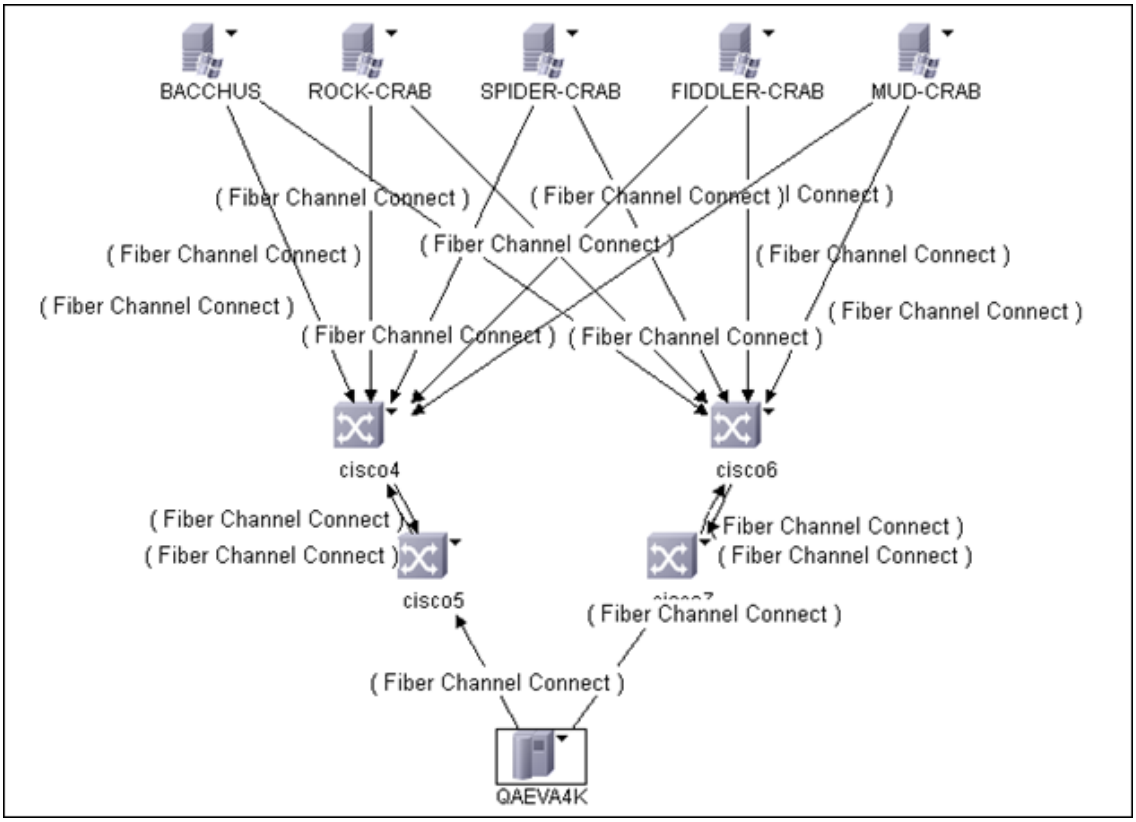
SANscreen integration supports version 5.1.2 (275) of NetApp SANscreen and version 6.2x of the product which has been renamed NetApp OnCommand Insight.

Topology

The following diagram illustrates the storage topology and shows the relationships between logical volumes on a storage array and those on servers:



The following diagram illustrates the SAN Topology and shows the fiber channel paths between storage arrays, switches, and servers:



How to Discover NetApp SANscreen

This task includes the following steps:

1. ### Prerequisite - Shell Connectivity

   Ensure there is Shell connectivity to one or more nodes of the SANscreen domain.

   For credential information, see "Supported Protocols" on page 82.

2. ### Modify the Classpath

   a. Navigate to **Admin > Discovery > Manage Discovery Resources**.

   b. Under **Discovery Resources**, expand **AutoDiscoveryContent > Configuration Files** and select **globalSettings.xml**. The XML content is displayed on the right.

   c. Add ";SANscreen/*.*" to the tag **<property name="AdditionalClasspath>** so it looks like:

   ```
   <property name="AdditionalClasspath">
   db/oracle/*.*;…;SANscreen/*.*</property>
   ```

   d. Restart the Probe(s).

3. ### Run the Discovery

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

   b. Run the **TCP Ports** job in order to discover SANscreen WebService ports.

4. ### Define the integration point

   In **Data Flow Management > Integration Studio**, define a new integration point:

   a. Provide a name and description

   b. Select the **NetApp SANscreen/On Command Insight** adapter and enter the required properties as follows:

Attribute	Description
ChunkSize	The number of CIs to pull from SANscreen/OnCommand Insight per query. The default is 1000.
Probe Name	Select the name of the Probe on which this integration will run.
Trigger CI Instance	Select the IpServiceEndpoint at which the SANscreen/OnCommand Insight service is running.

   A predefined job appears by default. Define a synchronization schedule if required. Jobs can also be run without a schedule.

   c. Save the job definition and then the integration point.

   d. Run a full synchronization for each job at least once.

SANscreen Adapter

Input CIT

IpServiceEndpoint

Input Query

Triggered CI Data

Name	Value
ip_address	${SOURCE.bound_to_ip_address}
port	${SOURCE.network_port_number}

Used Scripts

SANscreen_Discovery.py

Discovered CITs

- **Composition**
- **Containment**
- **Cpu**
- **Dependency**
- **Fiber Channel Connect**
- **Fibre Channel HBA**
- **Fibre Channel Port**
- **Fibre Channel Switch**
- **IpAddress**
- **LogicalVolume**
- **Membership**
- **Node**
- **Storage Array**

- **Storage Processor**
- **Unix**
- **Windows**

Global Configuration Files

None

Parameters

ChunkSize

Default: 1000

SANscreen Integration by WebServices Job

Adapter

This job uses the **NetApp SANscreen/OnCommand** integration adapter.

Parameters

ChunkSize

Default: 1000

Integration Flow

The adapter works as follows:

1. Connect to the SANscreen WebService API using credentials from the SANscreen protocol.
2. Query for storage arrays and create STORAGE ARRAY CIs.
3. Query for logical volumes, fiber channel adapters (Host Bus Adapters), and fiber channel ports on each storage array and create LOGICAL VOLUME, HBA, and FC PORT CIs.
4. Query for fiber channel switches and create FC SWITCH CIs.
5. Query for fiber channel adapters and ports on each fiber channel switch and create HBA and FC PORT CIs.
6. Query for hosts/servers and create appropriate COMPUTER, WINDOWS, or UNIX CIs.
7. Query for logical volumes, fiber channel adapters (Host Bus Adapters), and fiber channel ports on each host/server and create LOGICAL VOLUME, HBA, and FC PORT CIs.
8. Query for paths between hosts/servers and storage arrays and add FCCONNECT relationships between respective hosts/servers, switches, and storage arrays.
9. Query for logical volume mapping between logical volumes on hosts/servers and logical volumes on storage arrays and add DEPEND relationships between respective hosts/servers and storage arrays.

Troubleshooting and Limitations

Problem: Depending on the version of NetApp SANscreen or OnCommand Insight in use, discovery may fail with the following error message in the Probe wrapper logs:

```
SANscreen: Internal error. Details: java.lang.ClassCastException:
com.sun.xml.messaging.saaj.soap.ver1_1.Message1_1Impl
```

Solution: In this case, replace **<DDM>\root\lib\collectors\discoveryProbe\discoveryResources\SANscreen\sanscreen_api.jar** with the corresponding file from the SANscreen server in use.

Chapter 53

Network Node Manager (NNMi) Integration

This chapter includes:

Overview

You integrate NNMi with UCMDB using the Data Flow Management (DFM) application.

When you activate the NNMi integration, DFM retrieves Layer 2 network topology data from NNMi and saves the data to the UCMDB database. Users can then perform change management and impact analysis.

Use Cases

This document is based on the following use cases:

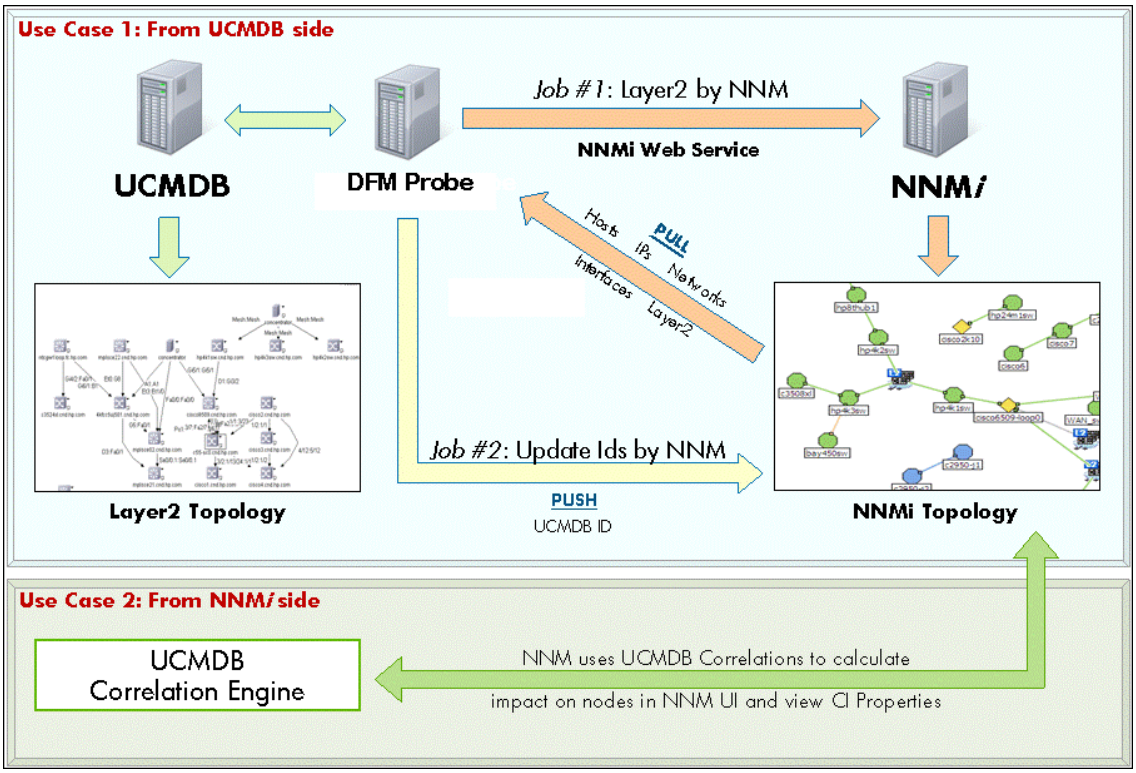
- **Use Case 1:** A UCMDB user wants to view the Layer 2 network topology supporting servers and applications. The requirement is to use NNMi as the authoritative source for that information with access through the Universal CMDB application.
- **Use Case 2:** An NNMi operator wants to view the impact of a network access switch infrastructure failure where the impact data is available in UCMDB. The NNMi operator selects an incident or a node in NNMi and then enters a request for impacted CIs.

Supported Versions

Out of the box, the following software versions are supported:

- Data Flow Probe version 9.02 or later
- HP NNMi version 8.1, 8.11, 9.0, 9.1

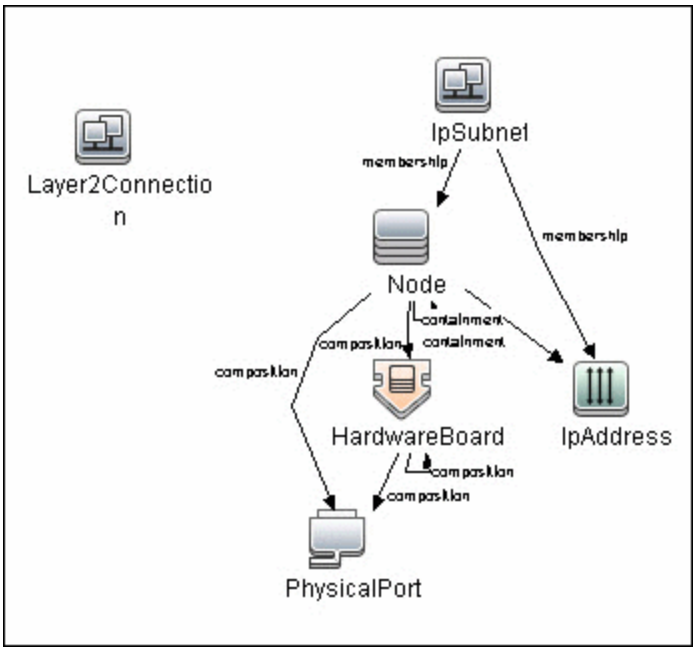
NNMi - UCMDB Integration Architecture



Topology

Layer2 by NNM Job

Note: For a list of discovered CITs, see "Discovered CITs" on page 682.



How to Run NNMi–UCMDB Integration

This task includes the steps to run the NNMi-UCMDB integration jobs.

Note: To avoid conflict, do not run the UCMDB Layer 2 discovery jobs when running the NNMi integration.

This task includes the following steps:

1. **Run NNMi Integration**

   In DFM, in the Integration Studio, create a new integration point.

   a. Provide a name and description for the integration point.

   b. Under **Integration Properties > Adapter**, select the appropriate adapter:

      - **Population from NNMi** Use this adapter to run population jobs. For more details see "How to Run NNMi–UCMDB Integration" above.
      - **Push IDs into NNMi** Use this adapter to run push jobs. For more details see "How to Run NNMi–UCMDB Integration" above.

   c. Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.

   d. Under **Adapter Properties > Trigger CI instance** select:

      - **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI; or
      - **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

      Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

   e. Under **Adapter Properties > Credentials ID** select the appropriate credentials for connection to the NNMi server.

   f. Save the Integration Point.

   g. Run the job.

   Note: For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

2. **Validate results**

   Verify that data was discovered using the NNMi integration jobs.

a. For the **NNMi population** job:
   - In UCMDB, navigate to **Admin > Modeling** > **IT Universe Manager.**
   - In the **CI Selector** pane, select **View Browser**.
   - In the **View** drop-down menu, select **Layer 2**. Select a view. The view displays the CIs and relationships discovered by the integration job.

b. For the **NNMi push** job:
   - In NNMi, open an NNMi node that was discovered in UCMDB.
   - On the **Custom Attributes** tab, look for the **UCMDB_ID** custom attribute. This attribute should contain the UCMDB ID of the corresponding host in UCMDB.

How to Manually Add the IpAddress CI of the NNMi Server

Note: When you installed HP Universal CMDB, you may have installed a bundled UCMDB that uses a Foundation license. If your UCMDB installation has a Foundation license deployed, use the steps in this section to manually add an **IpAddress** CI. If any other license (Basic or Advanced) is deployed on the UCMDB server, discover the IPAddress CI as described in "How to Run NNMi–UCMDB Integration" on page 676.

To manually add the IpAddress CI of the NNMi server

1. Verify that the Data Flow Probe is correctly installed and connected to the UCMDB Server.
2. Add the IP of the NNMi server to the Data Flow Probe range:

   In the **Data Flow Probe Setup** module, select the Probe that is to be used for the NNMi integration, and add the IP address of the NNMi server to its range. For details, see "Add/Edit IP Range Dialog Box" in the HP Universal CMDB Data Flow Management Guide.
3. Insert the **Address** CI of the NNMi server in the CMDB:
   a. In **Modeling** > **IT Universe Manager,** in the CI Selector pane, click the **Browse Views** tab and select **Network Topology** from the **View** drop-down menu.
   b. Click the **New CI** button.
   c. In the New CI dialog box, select the **IpAddress** CIT from the tree and enter the following values:

Field	Description
IP Address	The IP address of the NNMi server.
IP Domain Name	The UCMDB domain name (for example, `DefaultDomain`).
IP Probe Name	The name of the Data Probe (for example, `DefaultProbe`).

   d. Save the **IpAddress** CI.

How to Set Up HP NNMi–HP UCMDB Integration

The following steps describe how to configure NNMi to communicate with UCMDB:

Configure the connection between NNMi and UCMDB

On the NNMi management server, do the following:

1. In the NNMi console, open the **HP NNMi–HP UCMDB Integration Configuration** form (**Integration Module Configuration > HP UCMDB**).
2. Select the **Enable Integration** check box to activate the remaining fields on the form.
3. Enter the information for connecting to the NNMi management server. For information about these fields, see NNMi Management Server Connection.
4. Enter the information for connecting to the UCMDB server. For information about these fields, see UCMDB Server Connection.
5. Click **Submit** at the bottom of the form.

   A new window displays a status message. If the message indicates a problem with connecting to the UCMDB server, re-open the **HP NNMi–HP UCMDB Integration Configuration** form (or press **ALT+LEFT ARROW** in the message window), and then adjust the values for connecting to the UCMDB server as suggested by the text of the error message.

Customize the integration

On the NNMi management server, do the following:

1. In the NNMi console, open the **HP NNMi–HP UCMDB Integration Configuration** form (**Integration Module Configuration > HP UCMDB**).
2. Enter values for the following fields:
   - HP UCMDB Correlation Rule Prefix
   - HP UCMDB Impact Severity Level (1–9)

   For details on these fields, see Integration Behavior.
3. Click **Submit** at the bottom of the form.

NNM Integration Job

Adapter Parameters

Parameter	Description
discoverDisabledIps	Defines whether the integration should discover disabled IPs. When set to false, the integration does not discover disabled IPs. **Default**: false.
discoverLayer2	Defines whether the integration should discover the Layer2Connection CIs from NNMi. When set to true, the integration fetches all the Layer2Connections-related data, iteratively querying for a specified number of Layer2Connections from NNMi (based on value of the **pageSizeLayer2** parameter), then querying for Network Interfaces on the ends of Layer2Connection and Nodes hosting these interfaces with instant push of collected topology to UCMDB. **Default:** true
discoverNodes	Defines whether the integration should discover all the Nodes that are registered in NNMi, regardless of their inclusion into Layer2 Topology or VLANs. When set to true, integration fetches all the Nodes with connected IpAddresses, Interfaces, HardwareBoards, Physical Ports and IpSubnets, iteratively querying for a specified number of Nodes with related data from NNMi (based on value of the **pageSizeNodes** parameter) and instantly pushing collected topology into UCMDB. **Default:** true
discoverNonManagedInterfaces	Defines whether the integration should discover non-managed interfaces. When set to false, the integration does not discover non-managed interfaces. **Default**: false.

Parameter	Description
discoverNonManagedNodes	Defines whether the integration should discover non-managed nodes. When set to false, the integration does not discover non-managed nodes. **Default**: false. When this parameter is set to true, **discoverDisabledIps** and **discoverNonManagedInterfaces** are ignored for non-managed nodes; the integration will discover everything, including disabled ips and non-managed interfaces.
discoverPhysicalPorts	Defines whether the integration should discover physical ports. When set to false, the integration does not discover physical ports. **Default**: false. When this parameter is set to false, the integration does not discover **VLANs** and **HardwareBoards**.
discoverVlans	Defines whether the integration should discover all the VLANs that are registered in NNMi. When set to true, integration fetches all the VLANs with member Physical Ports, Hardware Boards and Nodes hosting those Physical Ports and Node-related topology, iteratively querying for a specified number of VLANs (based on the value of pageSizeVlans parameter), getting all the necessary related topology and instantly reporting it back to UCMDB. **Default:** true This parameter is ignored if **discoverPhysicalPorts** is set to false.
pageSizeLayer2	Defines the number of Layer2Connection CIs to fetch from NNMi per one query. **Default:** 5
pageSizeNodes	Defines the number of Nodes to fetch from NNMi per one query. **Default:** 10
pageSizeVlans	Defines the number of VLANs to be queries from NNMi per one query. **Default:** 1

Discovered CITs

- **Composition**
- **Containment**
- **HardwareBoard**
- **IPAddress**
- **IpSubnet**
- **Layer2Connection**
- **Membership**
- **Node**
- **PhysicalPort**
- **Realization**

Note: To view the topology, see "Topology" on page 675.

How to Customize Integration

This section describes how to customize the NNM Integration package in order to report additional attributes or entities, or to perform other changes.

Included Scripts

The NNM Integration package includes the scripts detailed in the following table:

Name	Description
nnmi_api.py	Implements API for accessing and retrieving data in NNM. Includes definition of base entities, such as **Node** or **Interface**, collections, topology and fetcher classes.
nnmi_filters.py	Support module used by **nnmi_api.py**. Includes definition of filters, allowing creation of advanced expressions when querying data in NNM. In general, you should not modify this script.
nnmi.py	Uses API provided by **nnm_api.py** to retrieve data, form topology into **ObjectStateHolder** objects and send result to UCMDB.
NNM_Integration.py	Main entry point of the integration. Contains **DiscoveryMain** method which is called by probe, and uses **nnmi.py**.
NNM_Integration_Utils.py	Previous version of NNM integration implementation, used by **CheckCred.py** script which validates credentials.
NNM_Integration_Utils_9.py	Previous version of NNM integration implementation for UCMDB 9.0
NNM_Update_Ids.py	Support module which updates custom attribute in NNM with UCMDB ID of the CI.

Customization Step by Step

The following steps illustrate customization of integration with an example, showing how you discover and report an additional attribute for an Interface. In this example, the attribute is the **managementMode** property in the NNM interface.

1. **Prerequisites**

   a. You need to know the **name** of the attribute in NNM. Here it is **managementMode**.

   b. You need to know the **name of a method** which should be called on the stub to read the corresponding property of entity, in this case Interface. This information is in the corresponding WSDL file or API documentation.

2. **Property Retrieval**

   For the new property to be reported, it should be retrieved first. To do that, you must modify the **nnmi_api.py** script.

   a. Locate Entity class

   Open **nnm_api.py** and find the corresponding Entity class. Entity class names follow the pattern **Nms*Entity**. For this example, you want **NmsInterfaceEntity**.

   nnmi_api.py

   ```
   class NmsInterfaceEntity(BaseNmsEntity):
       ...
   ```

   b. Add new entry to **field_names** tuple

   Add a new field to the **field_names** tuple. This tuple is used for real-time entity introspection, mainly debugging.

   Note: For use in Python classes, translate a camel cased field name into an underscore separated field name. For example: **management_mode**.

   nnmi_api.py

   ```
   field_names = (
       'id',
       'name',
       'hosted_on_id',
       'connection_id',
       'if_index',
       'if_alias',
       'if_descr',
       'if_name',
       'if_speed',
       'physical_address',
       'if_type',
       'uuid',
       'management_mode',
   )
   ```

c. Modify constructor of the Entity to read the new property

Modify **__init__ method** of the entity to read the new property. Modification depends on whether you need property post-processing.

i. Simple read

Call the method on the stub.

nnmi_api.py

```
class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type          = item.getIfType()
        self.uuid             = item.getUuid()

        self.management_mode = item.getManagementMode()
```

ii. Read with post processing

Delegate reading of the property to a new method which performs post processing for the retrieved value. In this case the new method is added to an **NmsInterfaceEntity** class called **_get_management_mode** which just strips the value of white spaces and makes it lowercase.

nnmi_api.py

```
class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type          = item.getIfType()
        self.uuid             = item.getUuid()

        self.management_mode = self._get_management_mode(item)
         ...
    def _get_management_mode(self, item):
        management_mode = item.getManagementMode()

        if management_mode:
            return management_mode.strip().lower()
        else:
            return ''
```

3. **Property reporting**

Once the modifications to **nnmi_api.py** are done, the new property is available, but it is not reported yet. To add reporting for this property, you need to modify the **nnmi.py** script.

a. Locate **Builder** class

Open the **nnmi.py** script and find the corresponding **Builder** class for the entity being reported. All **Builder** names match the ***Builder** pattern. In your case, you need the **InterfaceBuilder** class.

```
nnmi.py
class InterfaceBuilder:
    ...
```

b. Locate a place where **ObjectStateHolder** is created

Each **Builder** has a build method and several other methods which assist in the creation of the **ObjectStateHolder**. Find where this **ObjectStateHolder** is created either directly or indirectly by calling the method in the **modeling.py** module. In your example, **ObjectStateHolder**is created in the method **_createInterfaceOsh**.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        return interfaceOsh
```

c. Add new attribute reporting

Now you add reporting of the new property. here, you report the new value to the attribute **interface_management_mode** of the **Interface** CIT which you previously created.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        interfaceOsh.setAttribute('interface_management_mode',
interface.management_mode)

        return interfaceOsh
```

Troubleshooting and Limitations

This section describes troubleshooting and limitations for NNMi Integration.

- **Problem:** The NNMi Web service responds with a **cannot interrogate model** message.

  Solution: This message usually indicates that the Web services request made to the NNMi server is incorrect or too complex to process. Check the NNMi jbossServer.log file for details.

- **Problem:** If an excessive number of nodes are to be updated with the same UCMDB ID, it may take a while for the update adapter to complete.

  Solution: The volume of data retrieved from the NNMi server might be large. The recommended memory requirements for the Data Probe process is 1024 MB. Since the NNMi Web service enables updating the individual nodes one at a time, the time to update the nodes may take a while.

- **Problem:** The **Layer 2 by NNM** job finishes with the following warning: `Failed to get any Layer 2 links from NNM.`

  Solution: Refer to technical article KM629927 on the HP support Web site at http://support.openview.hp.com.

- **Problem:** Either of the NNMi integration jobs fails with the following error in the DFM log files: `com.hp.ov.nms.sdk.node.NmsNodeFault: Cannot interrogate model.`

  Solution: This error typically means that the NNMi server failed to process the Web services call. Check the following two logs on the NNMi server for exceptions when the integration was activated:

  - jbossServer.log
  - sdk.0.0.log

- **Problem:** Either of the NNMi integration jobs fail with the following error: `Could not find Discovery Probe 'DefaultProbe'. Task for TriggerCI will not be created.`

  Solution:

  a. Right-click the job and select **Go To Adapter**.

  b. Click the **Adapter Management** tab.

  c. Select the **Override default Probe selection** check box, and enter the name of the Probe used for the NNMi integration in the **Probe** field.

  d. Click **Save** to save the adapter, then reactivate the job against the **IpAddress** CI of the NNMi server.

- **Problem:** NNMi integration fails with error: `NNM integration: Job fails with "nested exception is: java.net.SocketException: Software caused connection abort: recv failed"`

  Solution:

a. Stop the Data Flow Probe.

b. Back up the following files:

<hp>\UCMDB\DataFlowProbe\content\lib\axis-1.4.jar

<hp>>\UCMDB\DataFlowProbe\content\lib\axis-jaxrpc-1.4.jar

<hp>\UCMDB\DataFlowProbe\content\lib\axis-saaj-1.4.jar

<hp>\UCMDB\DataFlowProbe\content\lib\axis-wsdl4j-1.5.1.jar

c. Delete the files described in step **b** from:

<hp>\UCMDB\DataFlowProbe\content\lib

d. Start the Data Flow Probe.

e. Rerun NNMi discovery.

Limitation

When integrating with multiple NNMi servers, each node (host) may only be managed by one NNMi server at the same time.

Chapter 54

ServiceNow Integration

This chapter includes:

Overview

This integration adapter provides the ability to push CIs and relationships from UCMDB to ServiceNow. The adapter uses an XML mapping framework that enables users to dynamically map CI Types between UCMDB and ServiceNow, without requiring code changes.

Supported Versions

This integration solution supports pushing CIs to ServiceNow from HP Universal CMDB version 9.02 and later.

How to Integrate ServiceNow with UCMDB

This task explains how to integrate ServiceNow with UCMDB.

1. **Configure queries**

   The CIs and relationships to be pushed to ServiceNow have to be queried from UCMDB using TQL queries. Create integration type queries to query the CIs and relationships that have to be pushed to ServiceNow.

   An example of such a query, **ServiceNowSampleQuery**, is included with this integration package, and can be viewed in the Modeling Studio. For details on viewing queries in the Modeling Studio, see the *HP Universal CMDB Modeling Guide*.

2. **Create XML mapping files**

   For every query created in the step above, create an XML mapping file with exactly the same name (case-sensitive) as the integration query in the following directory:

   ```
   <UCMDB>\UCMDBServer\runtime\fcmdb\
   CodeBase\ServiceNowPushAdapter\mappings
   ```

   A sample mapping file for this integration, **ServiceNowSampleMapping.xml**, is provided out of the box with the package.

   For more information about mapping files, see the *HP Universal CMDB Developer Reference Guide*.

3. **Create the integration point**

   a. In Data Flow Management, in the Integration Studio, define a new integration point:

      i. Provide a name and description.

      ii. Ensure the **Is Integration Activated** option is enabled.

      iii. Under **Integration Properties > Adapter**, select **Service-Now Push**.

      iv. Under Adapter Properties:

Field	Value
Service-Now Domain	Domain name used to access the ServiceNow instance. Usually **service-now.com**.
Service-Now Instance	The ServiceNow instance being used. For the demonstration instance, enter **demo**.
Port	Default 443, for HTTPS.
Protocol	HTTP or HTTPS.
Proxy Server Name/IP	If an HTTP(S) proxy service is used to access the Internet, enter the proxy server name.
Proxy Server Port	HTTP(S) proxy server port.
Credentials	Define the ServiceNow instance username and password in the Generic Protocol. For the demonstration website, use **admin/admin**. For credential information, see Supported Protocols in the *HP Universal CMDB Discovery and Integration Content Guide*
Probe Name	Select the name of the Probe on which this integration will run.

b. Test the connection to the target CMDB server. If the connection fails, verify that the information provided is correct.

c. Save the integration point.

d. Add a new job definition to the integration point. Provide a name for the job definition and select the queries to use to synchronize data from UCMDB to ServiceNow. Define a synchronization schedule if required.

> **Note:** Jobs can also be run without a schedule.

e. Save the job definition, and then the integration point.

f. Run a full synchronization for each job at least once.

Integration Mechanism

The components responsible for the ServiceNow integration are bundled in the ServiceNow Integration package, **ServiceNow_Integration.zip**.

The integration mechanism works as follows:

1. **The Integration job queries UCMDB for CIs and relationships:**

   When an ad-hoc job is run from the integration point in the Integration Studio, the integration receives the names of the integration queries defined in the job definition, for that integration point.

   It queries UCMDB for the results of these queries (new, updated and deleted CIs and relationships) and then applies the mapping transformation according to the pre-defined XML mapping files for every TQL query.

   It then pushes the data to the Data Flow Probes.

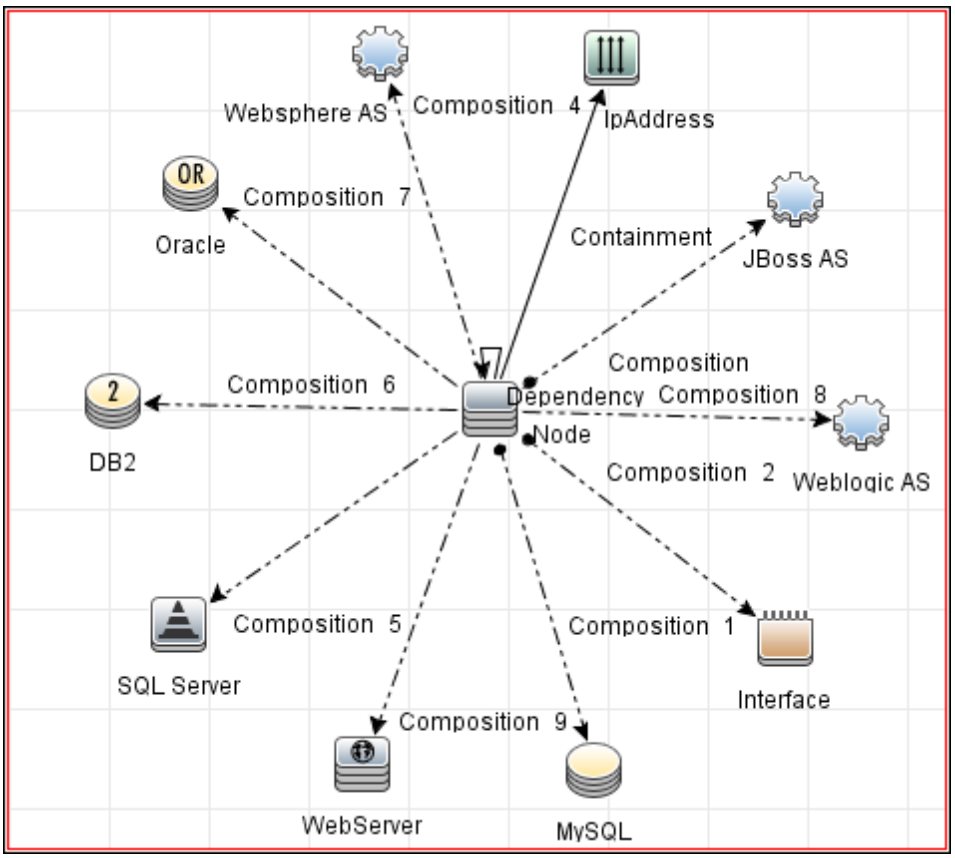
2. **The integration job sends the data to ServiceNow:**

   Next, on the Data Flow Probe side, the integration process receives the CI and relationship data sent from the UCMDB server, connects to the ServiceNow server using the Direct Web Services SOAP API, and transfers the CIs and relationships.

   Since the ServiceNow coalescing (CI reconciliation) mechanism is not available for the Direct Web Services API, a mapping of UCMDB CI IDs to ServiceNow SysIds is maintained on the discovery Probe. This mapping is used to update and delete CIs and relationships in ServiceNow.

Sample Integration Push Query

The following image displays a sample query to be pushed to ServiceNow:



Supported CITs

The following CIs and their relationship are supported:

- Apache Tomcat
- Apache
- DB2
- ESX Server
- Host
- IIS Web Server
- Interface
- Ip Address
- JBoss AS
- MySQL
- Net Device
- Oracle
- SQL Server
- Switch
- Unix Server
- Weblogic AS
- WebSphere AS
- Windows Server

Pushing Additional CITs

To push additional CITs, these CITs should be added to the mapping file. For details, see the *HP Universal CMDB Developer Reference Guide*.

Pushing additional CI Types requires corresponding JAR files to be generated using the WSDL URL for each CI Type. The WSDL URL can be generated using information from the **Direct Web Services section** at:

http://wiki.service-now.com/index.php?title=SOAP_Web_Service.

The resulting JAR files should be placed in the following directory on the UCMDB Data Flow Probe server:

<hp>\UCMDB\DataFlowProbe\runtime\probeManager\discoveryResources\Service-Now

Note: JAR files should be re-generated following CI Type updates in ServiceNow.

JAR files may be generated using WSDL2JAVA or other similar utilities. An example using this utility is at :

http://roseindia.net/webservices/axis2/axis2-client.shtml.

For CIs containing reference fields, the target data type of the attribute being mapped to a reference field should be set to the name of the reference table. For example, to populate the **Manufacturer** field on a **Windows Server** CI, the data type should be the reference table name **core_company** as shown below:

```
<source_ci_type name="nt" nameSpace="" query="">
    <target_ci_type name="cmdb_ci_win_server">
        <targetprimarykey>
            <pkey>ID</pkey>
        </targetprimarykey>
        <target_attribute name="virtual" datatype="boolean" >
            <map type="direct" source_attribute="host_isvirtual" />
        </target_attribute>
        <target_attribute name="category" datatype="String" >
            <map type="direct" source_attribute="os_family" />
        </target_attribute>
        <target_attribute name="short_description" datatype="String" >
            <map type="direct" source_attribute="discovered_description" />
        </target_attribute>
        <target_attribute name="correlation_id" datatype="String" >
            <map type="direct" source_attribute="global_id" />
        </target_attribute>
        <target_attribute name="model_number" datatype="String" >
            <map type="direct" source_attribute="discovered_model" />
        </target_attribute>
        <target_attribute name="name" datatype="String" >
            <map type="direct" source_attribute="display_label" />
        </target_attribute>
        <target_attribute name="os" datatype="String" >
            <map type="direct" source_attribute="discovered_os_name" />
        </target_attribute>
        <target_attribute name="model_id" datatype="cmdb_model" >
            <map type="direct" source_attribute="discovered_model" />
        </target_attribute>
        <target_attribute name="manufacturer" datatype="core_company" >
            <map type="direct" source_attribute="discovered_vendor" />
        </target_attribute>
    </target_ci_type>
</source_ci_type>
```

Troubleshooting and Limitations

This section describes troubleshooting and limitations for the ServiceNow-UCMDB integration.

- **Limitation:** The integration mapping file only allows mapping concrete CITs and relationships to the CITs and relationships in ServiceNow. That is, a parent CIT cannot be used to map its children CIs.
- **Limitation:** Since this adapter uses the ServiceNow Direct Web Services API, which does not support CI coalescing (reconciliation), if some CIs being pushed from UCMDB are already present in the ServiceNow CMDB, before the integration with UCMDB is installed, and if those CIs are (a) also in UCMDB; and (b) pushed into ServiceNow by the integration, those CIs are duplicated. (This is because UCMDB does not know these CIs are already in the ServiceNow CMDB.) After the adapter is installed, UCMDB keeps track of the CIs it pushes to ServiceNow, to prevent duplication.
- **Limitation:** ServiceNow Web Service Import Sets are currently not supported.

Chapter 55

Storage Essentials (SE) Integration

This chapter includes:

Overview

Integration involves synchronizing devices, topology, and the hierarchy of a customer storage infrastructure in the Universal CMDB database (CMDB). This enables Change Management and Impact Analysis across all business services mapped in UCMDB from a storage point of view.

You integrate SE with UCMDB using Data Flow Management (DFM).

When you activate the Storage Essentials integration, DFM retrieves data from the SE Oracle database and saves CIs to the Universal CMDB database. Users can then view SE storage infrastructure in UCMDB.

The data includes information on storage arrays, fibre channel switches, hosts (servers), storage fabrics, logical volumes, host bus adapters, storage controllers, and fibre channel ports. Integration also synchronizes physical relationships between the hardware, and logical relationships between logical volumes, storage zones, storage fabrics, and hardware devices.

Supported Versions

The integration procedure supports DFM version 9.02 or later and SE versions 6.x, 9.4, 9.41 and 9.5.

How to Perform the SE Integration

This task includes the steps to perform SE-UCMDB integration.

1. **Run the job - UCMDB 9.04 and later**

   For details on running integration jobs, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

   In DFM, in the Integration Studio, create a new integration point.

   a. Provide a name and description for the integration point.

   b. Under **Integration Properties > Adapter**, select the **Storage Essentials** adapter.

   c. Under **Adapter Properties > Probe Name** select the **Data Flow Probe**.

   d. Under **Adapter Properties >Trigger CI instance** select:

      i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or

      ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

   Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.

   e. Verify the credentials for the chosen CI instance. Right-click **Trigger CI instance** and select **Actions > Edit Credentials Information**.

f. Save the integration point.

g. Run the job.

2. **Run the job - UCMDB 9.03 and 9.02**

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

a. Prerequisite - set up protocol credentials.

- This integration uses the Generic DB Protocol (SQL).

> **Note:** For credential information, see "Supported Protocols" on page 82.

b. In DFM, in the Discovery Control Panel window, run one of the following sets of jobs to trigger SE discovery:

Set 1:

- **Range IPs by ICMP**. Discovers the IP address of the SE server.
- **Host Connection by Shell/WMI/SNMP**. Discovers operating system information on the SE server.
- **Host Resources and Applications by Shell/SNMP/WMI.** Discovers the Oracle database instance used by SE.
- **Oracle Database Connections by SQL.** Discovers Oracle databases using the Generic DB Protocol (SQL).

Set 2:

- **Range IPs by ICMP.** Discovers the IP address of the SE server.
- **Database TCP ports.**
- **Oracle Database Connections by SQL.** Discovers Oracle databases using the Generic DB Protocol (SQL).

c. Run the **SE Integration by SQL** job to discover storage infrastructure.

Storage Essentials Integration Packages

The integration includes two UCMDB packages:

- **SE_Discovery.zip**. Contains the trigger query for SE discovery, discovery script, adapter, and job.
- **Storage_Basic.zip**. Contains the new CI Type definitions, views, reports, and impact analysis rules. This package is common to all Storage Management integration solutions.

> **Tip:** You can include the SE job in the DFM schedule.
>
> **UCMDB 9.04 and later**: for details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

UCMDB 9.03 and 9.02: for details, see "Discovery Scheduler Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

Adapter Parameters

This job runs queries against Oracle materialized views that are installed and maintained by Storage Essentials in the Oracle database. The job uses a database CI as the trigger.

A switch or server in SE inherits from a Node CIT in UCMDB based on the following adapter parameters:

Parameter	Description
allowDNSLookup	If a node in the SE database does not have an IP address but has a DNS name, it is possible to resolve the IP address by the DNS name. **True:** If a node does not have an IP address, an attempt is made to resolve the IP address by DNS name (if a DNS name is available). **Default:** False
ignoreNodesWithoutIP	Defines whether or not nodes in SE without IP addresses should be pulled into UCMDB. • **True**. Nodes without IPs are ignored. • **False.** A Node CI is created with an SE ID as the node key attribute. **Note:** Setting this parameter to **False** may result in duplicate CIs in the CMDB. **Default:** True

Discovered CITs and Relationships

This section describes SE storage entities in UCMDB:

- **Fibre Channel Connect**. Represents a fibre channel connection between fibre channel ports.
- **Fibre Channel HBA**. Has change monitoring enabled on parameters such as state, status, version, firmware version, driver version, WWN, and serial number. A Fibre Channel HBA inherits from the Node Resource CIT.
- **Fibre Channel Port**. Has change monitoring enabled on parameters such as state, status, WWN, and trunked state. Since a Fibre Channel Port is a physical port on a switch, it inherits

from the Physical Port CIT under the NodeElement Resource CIT.

- **Fibre Channel Switch**. Falls under the Node CIT because SE maintains an IP address for each switch. Parameters such as status, state, total/free/available ports, and version are change monitored.

  This package retrieves Fibre Channel Switch details from the **mvc_switchsummaryvw** and **mvc_switchconfigvw** views. The job retrieves detailed information about Fibre Channel Ports on each switch from the **mvc_portsummaryvw** view.

- **Logical Volume**. Represents volumes on Storage Arrays and hosts with change monitoring on availability, total/free/available space, and storage capabilities.

- **Storage Array**. Represents a Storage Array with change monitoring on details such as serial number, version, and status. Since a storage array may not have a discoverable IP address, it inherits from the Network Device CIT.

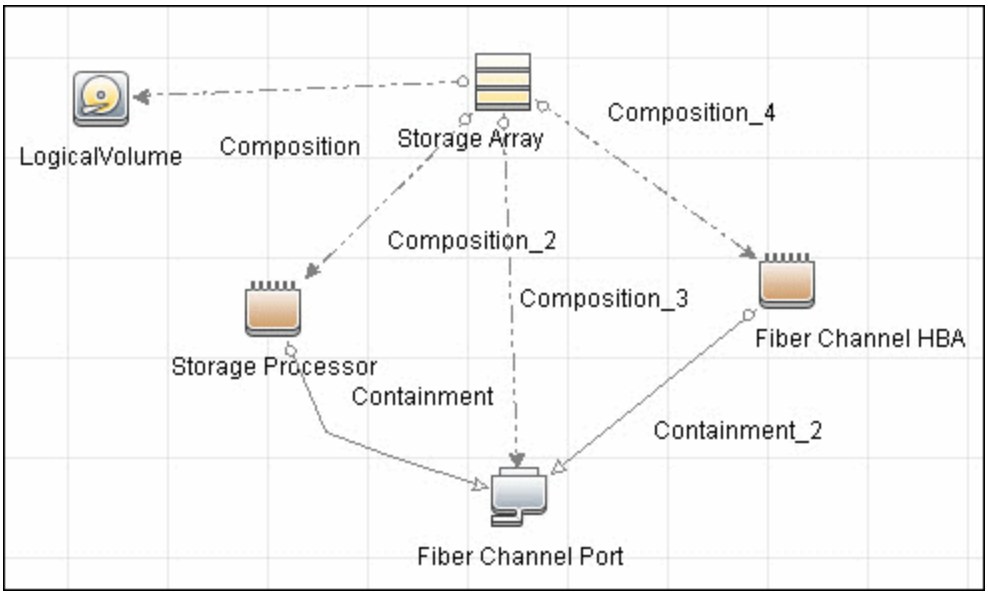
  This CIT retrieves Storage Array details from the **mvc_storagesystemsummaryvw** view. DFM retrieves detailed information on Storage Processors and HBAs from the **mvc_ storageprocessorsummaryvw** and **mvc_cardsummaryvw** tables respectively.

  The SE database may possibly not be able to obtain IP address information on Storage Arrays for a variety of technical and policy related reasons. Since a Storage Array is a host as far as DFM is concerned, DFM assumes that the serial number of a Storage Array is unique and uses this as the primary key. The CI is then manually set as a complete host. If the serial number of a Storage Array is not available, the array is discarded.

  Since Fibre Channel Ports may be present on a Storage Array, Storage Processor, or HBA, DFM uses three separate queries to retrieve Fibre Channel Ports for each Storage Array. Detailed information about Fibre Channel Ports on each array are retrieved from the **mvc_ portsummaryvw** view. Since this view uses a container ID as the key, DFM queries the view by container ID for each Storage Array, each Storage Processor on a Storage Array, and each HBA on a Storage Array.

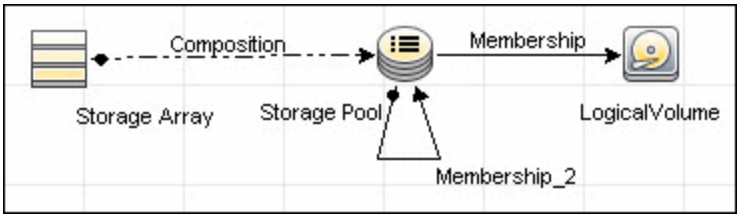
  DFM retrieves detailed information about Logical Volumes on each Storage Array from the **mvc_ storagevolumesummaryvw** view.

Results from these queries populate a map as shown below:



- **Storage Fabric**. Inherits from the Network Resource CIT and represents a storage fabric. This CIT has no change monitoring enabled.
- **Storage Processor**. Represents other storage devices such as SCSI controllers, and inherits from the Host Resource CIT. A Storage Processor CIT monitors change on parameters such as state, status, version, WWN, roles, power management, and serial number.
- **Storage Pool.** Storage Pool information is also collected from each Storage Array using the query below.

  Results from this query populate a map as shown below:



Node Details

DFM retrieves Host details from the **mvc_hostsummaryvw** view and detailed information on HBAs from the **mvc_cardsummaryvw** view.

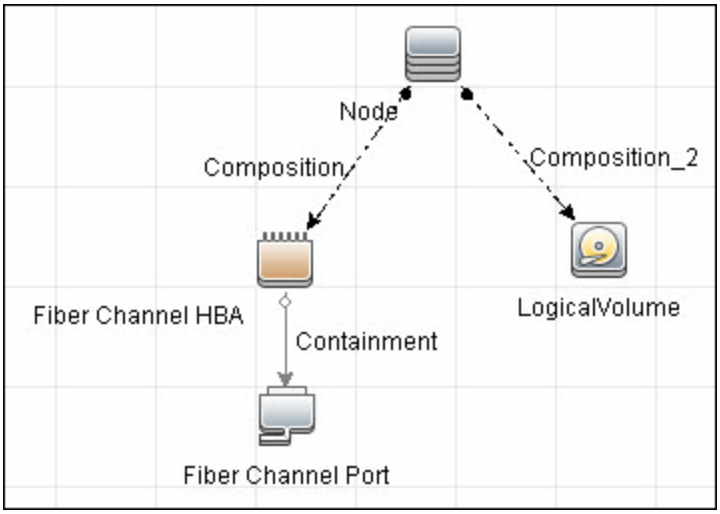
SE maintains information on Operating Systems, IP address, and DNS name on each host. DFM uses this information to create Node CIs (UNIX or Windows) and IpAddress CIs.

Since UCMDB uses the IP address of a node as part of its primary key, DFM attempts to use the IP address from SE for this purpose. If an IP address is not available, DFM then attempts to resolve the hosts IP address using a DNS name. If neither an IP address nor a DNS name is available, DFM ignores the host (see "Adapter Parameters" on page 699.

Similar to Storage Arrays, a node may have Fibre Channel Ports directly associated with itself or on HBAs on the host. The DFM job uses three separate queries to retrieve Fibre Channel Ports for each host. The job retrieves detailed information about Fibre Channel Ports on each host from the **mvc_portsummaryvw** view. Since this view uses a ContainerID attribute as the key, the job queries the view by containerID for each host, and each HBA on a host.

Finally, DFM retrieves detailed information about Logical Volumes on each host from the **mvc_hostvolumesummaryvw** and **mvc_hostcapacityvw** views. The **mvc_hostcapacityvw** view maintains capacity information for each volume over multiple instances in time, and the job uses only the latest available information.

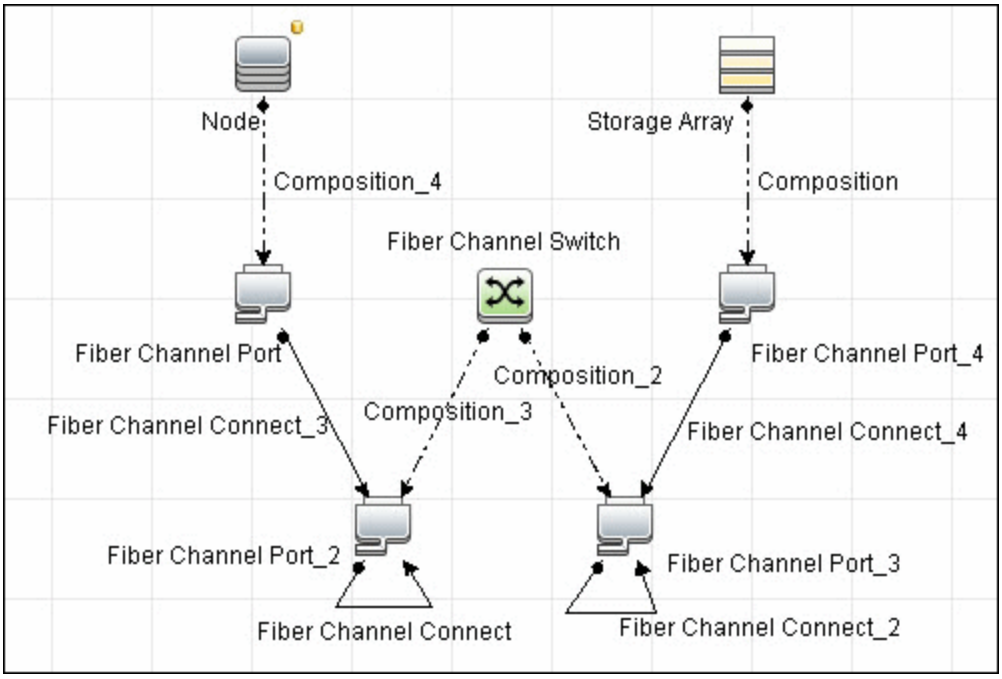
Results from these queries populate a map as shown below:



SAN Topology

SAN Topology consists of the Fibre Channel network topology and includes (fibre channel) connections between Fibre Channel Switches, Hosts, and Storage Arrays. SE maintains a list of WWNs that each Fibre Channel Port connects to, and this package uses this list of WWNs to establish Fibre Channel Connection links.

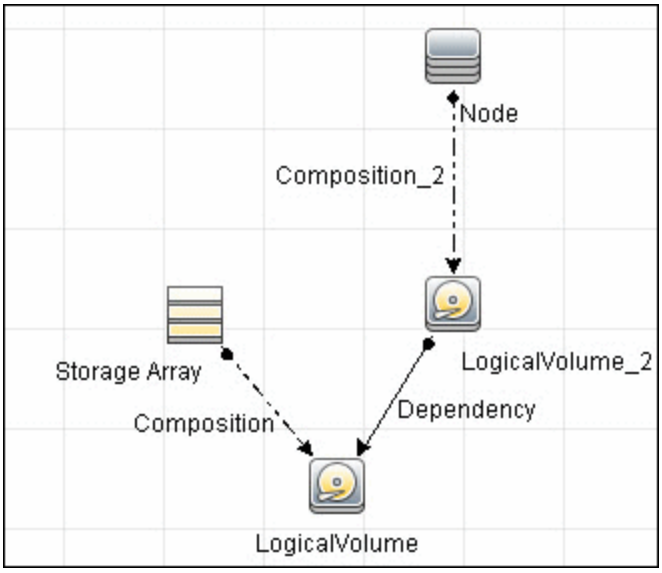
Results from these queries populate a map as shown below:



Storage Topology

Storage topology consists of relationships between Logical Volumes on a host and Logical Volumes on a Storage Array. DFM uses multiple tables to identify this relationship as shown in the query below. This view is a summary of all of the above information.

Results from these queries populate a map as shown below:



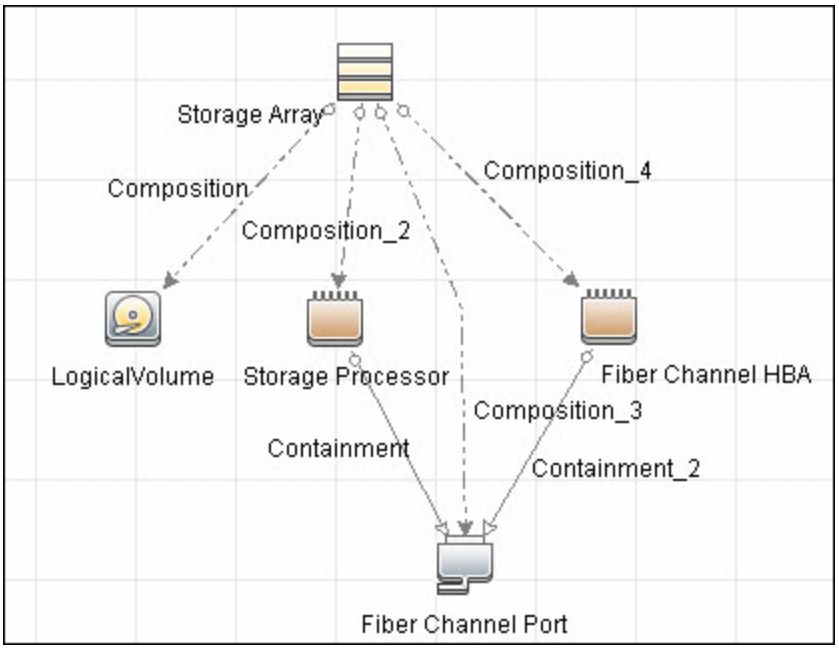
Views

The SE package contains views that display common storage topologies. These are basic views that can be customized to suit the integrated SE applications.

Storage Array Details

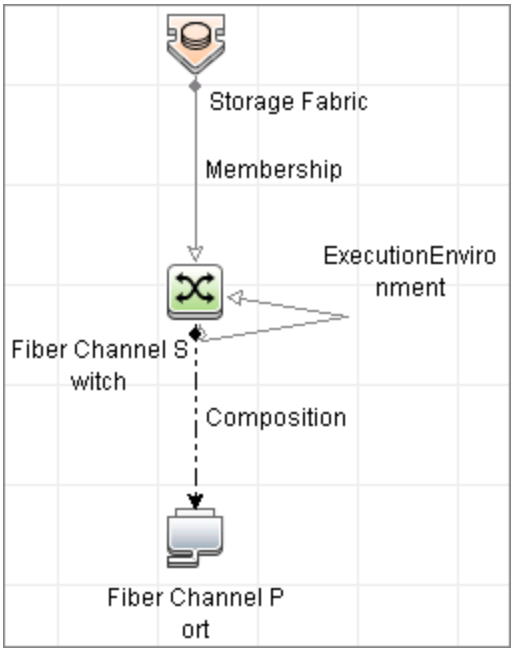
This view shows a Storage Array and its components including Logical Volumes, HBAs, Storage Processors, and Fibre Channel Ports. The view shows each component under its container Storage Array and groups Logical Volumes by CI Type.

Storage Array does not require all components in this view to be functional. Composition links stemming from the Storage Array have a cardinality of zero-to-many. The view may show Storage Arrays even when there are no Logical Volumes or Storage Processors.



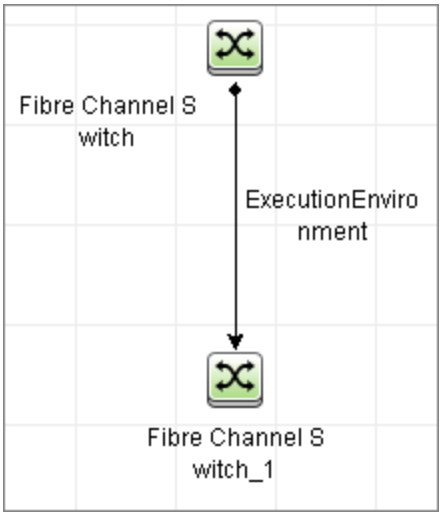
FC Switch Details

This view shows a Fibre Channel Switch and all connected Fibre Channel Ports.



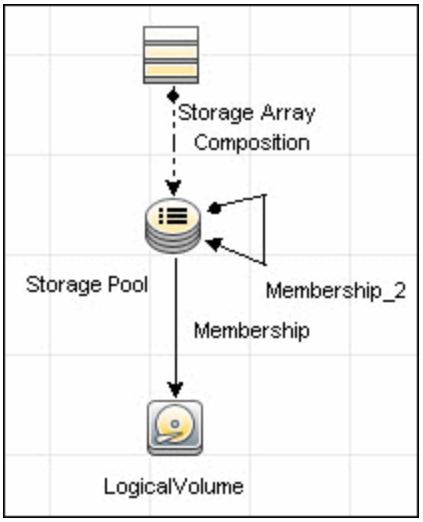
FC Switch Virtualization

FC Switch Virtualization consists of a physical switch or chassis, partitioned into multiple logical switches. Unlike Ethernet virtualization, physical ports are not shared among multiple virtual switches. Rather, each virtual switch is assigned one or more dedicated physical ports that are managed independently by the logical switches.



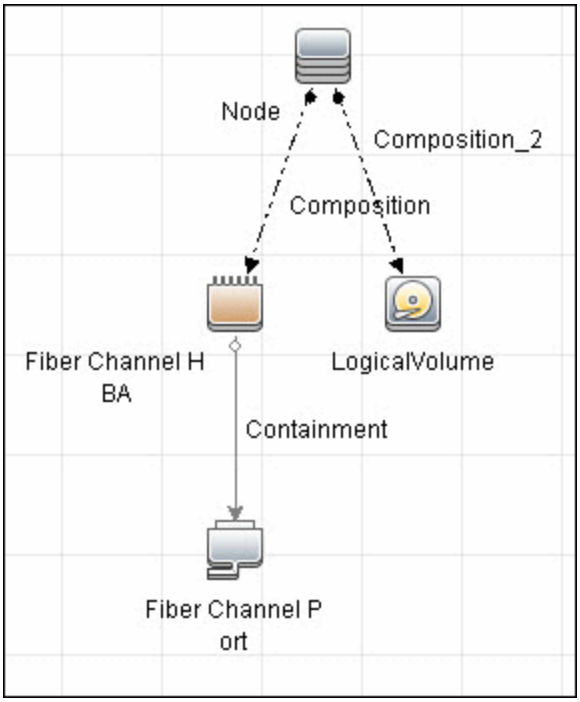
Storage Pool Details

This view shows Storage Pools with associated Storage Arrays and Logical Volumes.



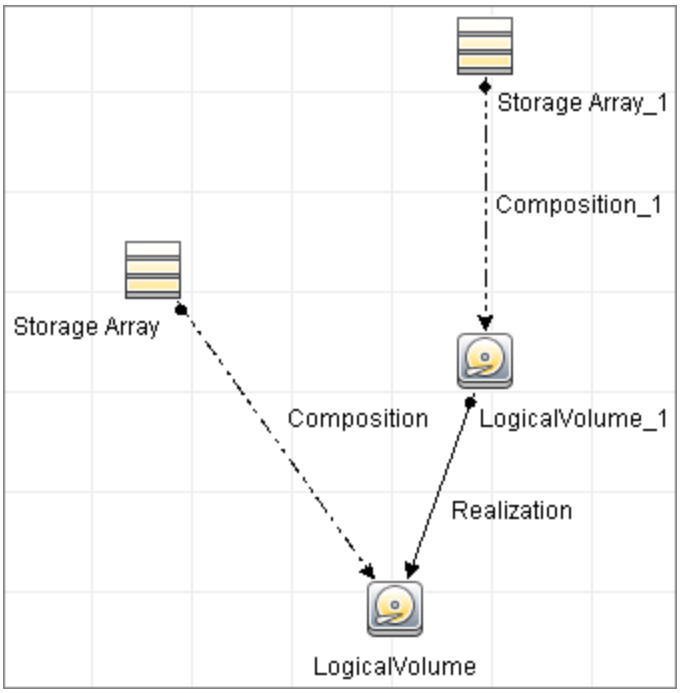
Host Storage Details

This view shows only Hosts that contain a Fibre Channel HBA or a Logical Volume. This keeps the view storage-specific and prevents hosts discovered by other DFM jobs from being included in the view.



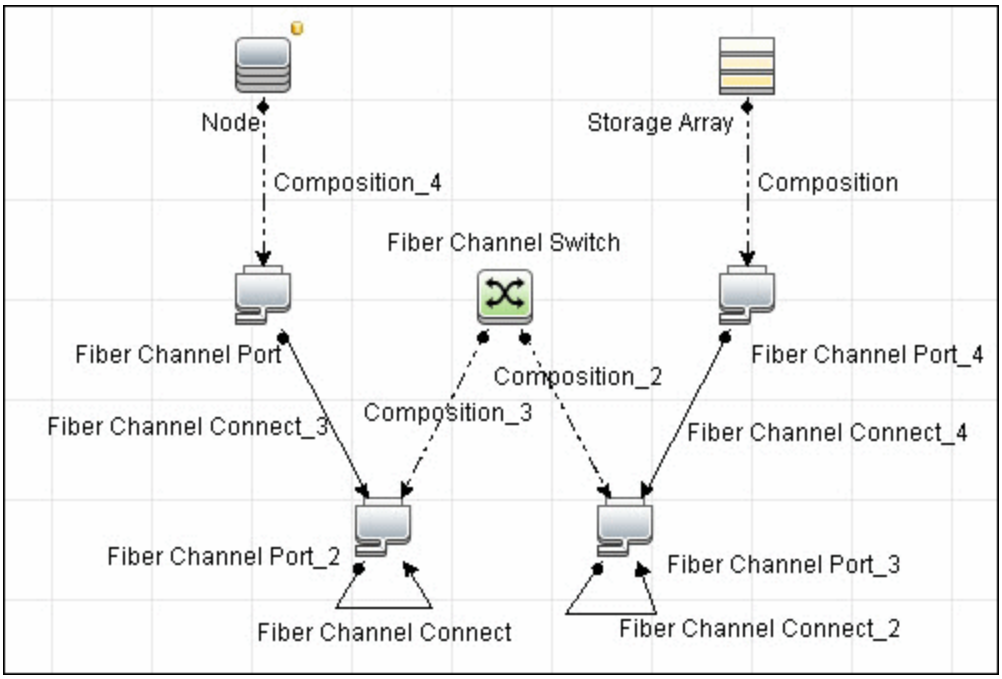
SAN External Storage

External storage configuration consists of a storage array presenting a logical volume that, in reality, belongs to another storage array. This is typically used in configurations where high-end, more expensive, front-end arrays present volumes from back-end, cheaper, storage to servers. The goal of this type of virtualization is to virtualize multiple disk arrays from different vendors, scattered over the network, into a single monolithic storage device that can be managed uniformly.



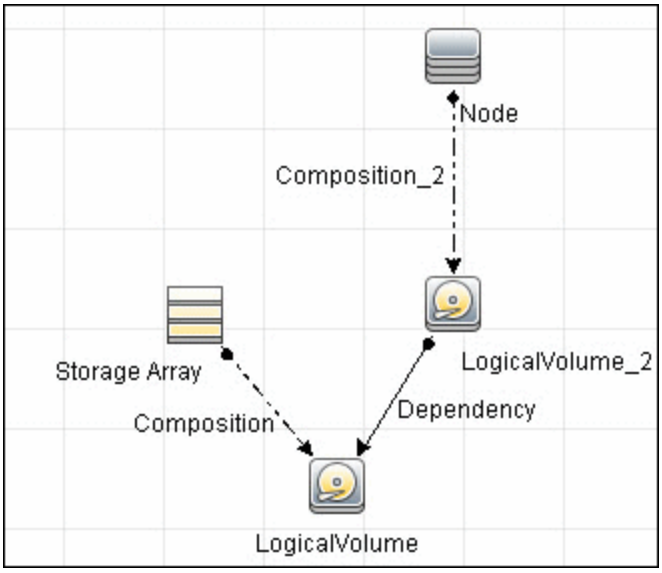
SAN Topology

This view maps physical connections between Storage Arrays, Fibre Channel Switches, and Hosts. The view shows Fibre Channel Ports below their containers. The view groups the Fibre Channel Connect relationship CIT to prevent multiple relationships between the same nodes from appearing in the top layer.



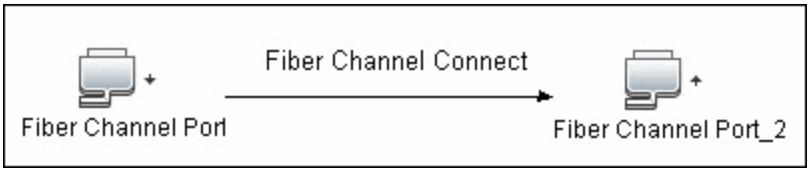
Storage Topology

This view maps logical dependencies between Logical Volumes on Hosts and Logical Volumes on Storage Arrays. There is no folding in this view.



FC Port to FC Port

This rule propagates events on a Fibre Channel Port to another connected Channel Port.



Example of HBA crashing on a Storage Array:

- The event propagates from the HBA to the Storage Array and the Logical Volumes on the Array because of the Storage Devices to Storage Array rule.
- The impact analysis event on the Logical Volume then propagates to other dependent Logical Volumes through the Logical Volume to Logical Volume rule.
- Hosts using those dependent Logical volumes see the event next because of the Host Devices to Host rule.
- Depending on business needs, you define impact analysis rules to propagate events from these hosts to applications, business services, lines of business, and so on. This enables end-to-end mapping and impact analysis using UCMDB.

Impact Analysis Rules

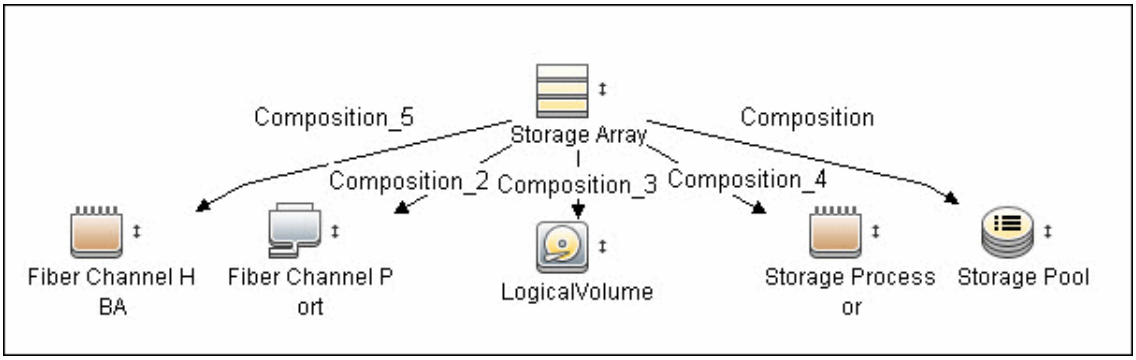
This package contains basic impact analysis rules to enable impact analysis and root cause analysis in UCMDB. These impact analysis rules are templates for more complex rules that you can define based on business needs.

All impact analysis rules fully propagate both Change and Operation events. For details on impact analysis, see "Impact Analysis Manager Page" and "Impact Analysis Manager Overview" in the HP Universal CMDB Modeling Guide.

Note: Impact analysis events are not propagated to Fibre Channel Ports for performance reasons.

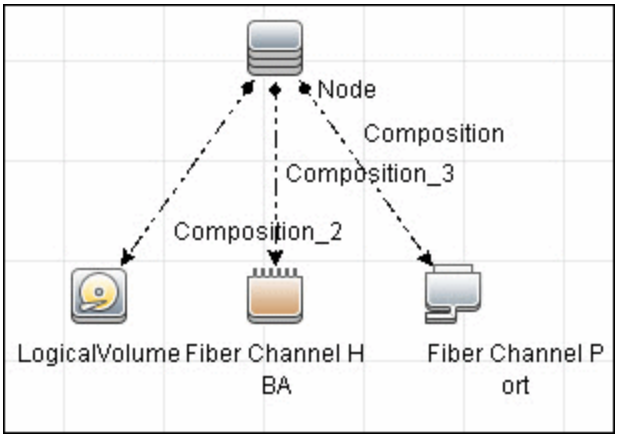
Storage Array Devices to Storage Array

This impact analysis rule propagates events between Logical Volumes, Storage Processors, Fibre Channel HBAs, and Storage Arrays.



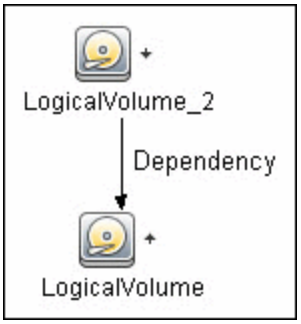
Host Devices to Host

This impact analysis rule propagates events between Fibre Channel HBAs and Hosts, and Logical Volumes on the Host.



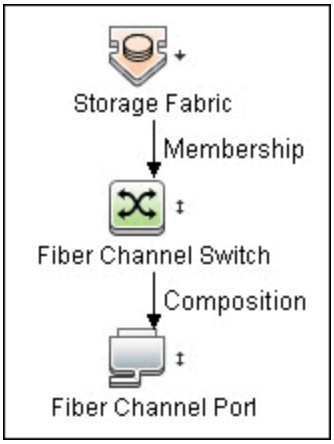
Logical Volume to Logical Volume

This impact analysis rule propagates events on a Logical Volume contained in a Storage Array to the dependent Logical Volume on the Host.



FC Switch Devices to FC Switch

This impact analysis rule propagates events from a Fibre Channel Port to and from a Switch. The event is also propagated to the associated Storage Fabric.



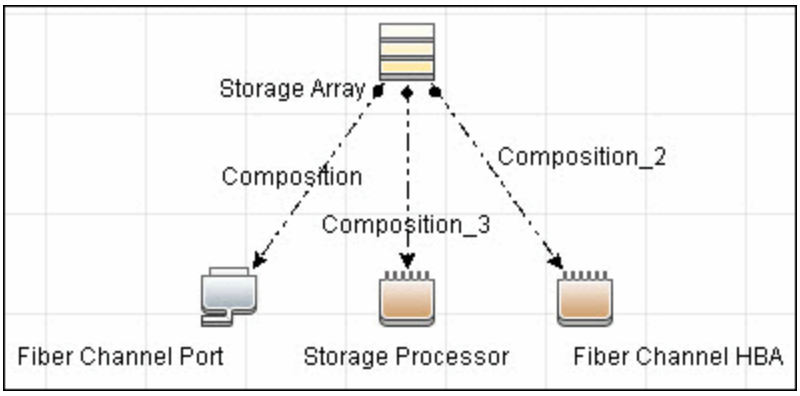
Reports

The SE package contains basic reports that can be customized to suit the integrated SE applications.

In addition to the system reports, Change Monitoring and Asset Data parameters are set on each CIT in this package, to enable Change and Asset Reports in Universal CMDB. For details see "Storage Array Configuration" below, "Host Configuration" on next page, "Storage Array Dependency" on next page, and "Host Storage Dependency" on page 717.

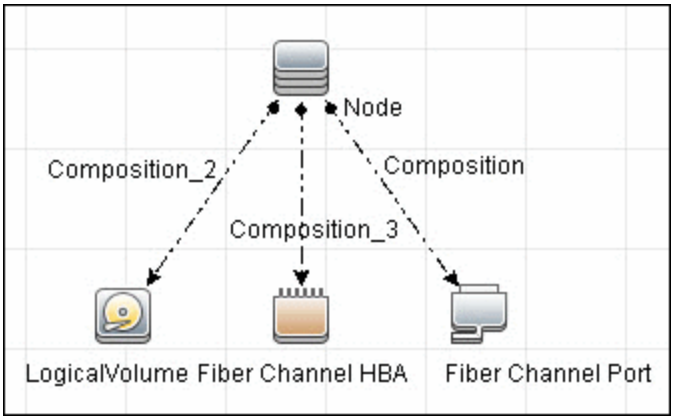
Storage Array Configuration

This report shows detailed information on Storage Arrays and its sub-components including Fibre Channel Ports, Fibre Channel Arrays, and Storage Processors. The report lists Storage Arrays with sub-components as children of the Array.



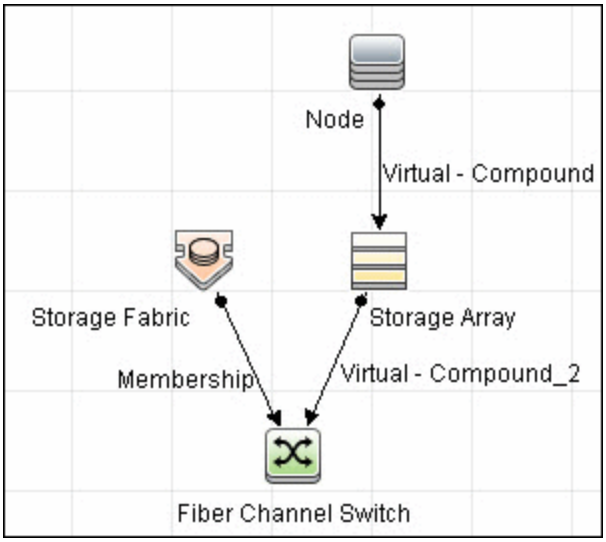
Host Configuration

This report shows detailed information on hosts that contain one or more Fibre Channel HBAs, Fibre Channel Ports, or Logical volumes. The report lists hosts with sub-components as children of the host.



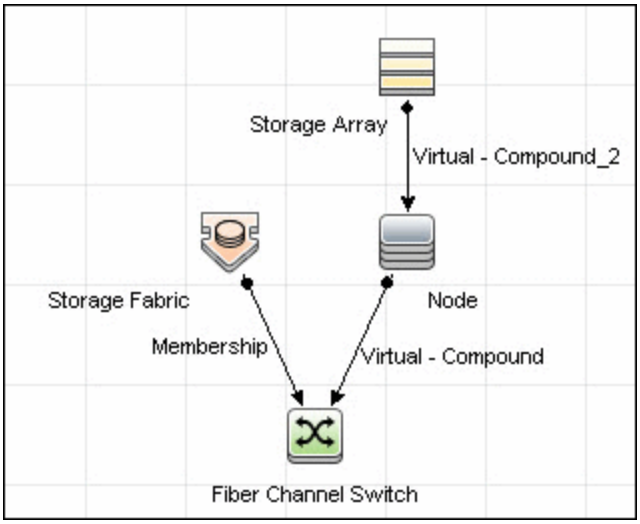
Storage Array Dependency

This report maps dependencies on a Storage Array. The report also displays information on switches connected to it.



Host Storage Dependency

This report shows detailed information on storage infrastructure dependencies of a Host. The report lists hosts and dependent components.



Storage Pool Configuration

This report shows detailed information on Storage Pool configuration.

Troubleshooting and Limitations

This section describes troubleshooting and limitations of Storage Essentials Integration.

- **Problem:** If the SE system has duplicate entries for nodes, switches or arrays, the job produces the following error message: "**Process validator error: multiple updates in bulk...**".

  Solution: This is expected behavior and does not affect population of valid CIs into UCMDB. To prevent this error message, duplicates must be removed from the SE system.

- **Error message:** "Please use the SE database with SID 'REPORT' for this integration."

  Solution: For integration with SE 9.4 or higher, you should configure the integration point for a REPORT database instance (and not for an APPIQ instance as applied for SE versions up to and including 6.3).

- **Limitation**: The credentials used for integration with SE should have access to the relevant materialized view status table as detailed below:

Storage Essentials	Materialized View Status Table
Version 6.03 and earlier	appiq_system.mview_status
Later than Version 6.03 and earlier than Version 6.1	appiq_system.mviewcore_status and appiq_system.mview_status
Later than Version 6.1 and earlier than Version 9.4	appiq_system.mview_module_status
Version 9.4 and later	appiq_system.mv_report_user_status

- **Limitation:** If the discovery does not find a valid IP address and serial number for a VMWare ESX server in the HP Storage Essentials database, that VMware ESX server is not reported to UCMDB.

Chapter 56

Troux Integration

This chapter includes:

Introduction

Troux is a leader in the EA (Enterprise Architecture) tools space. EA tools allow business users to understand the gaps between business demands and initiatives. Reviewing how your fixed budget aligns to business capabilities and how your discretionary spending is allocated across initiatives. Future-state scenario investigation can be accomplished prior to locking down your roadmap.

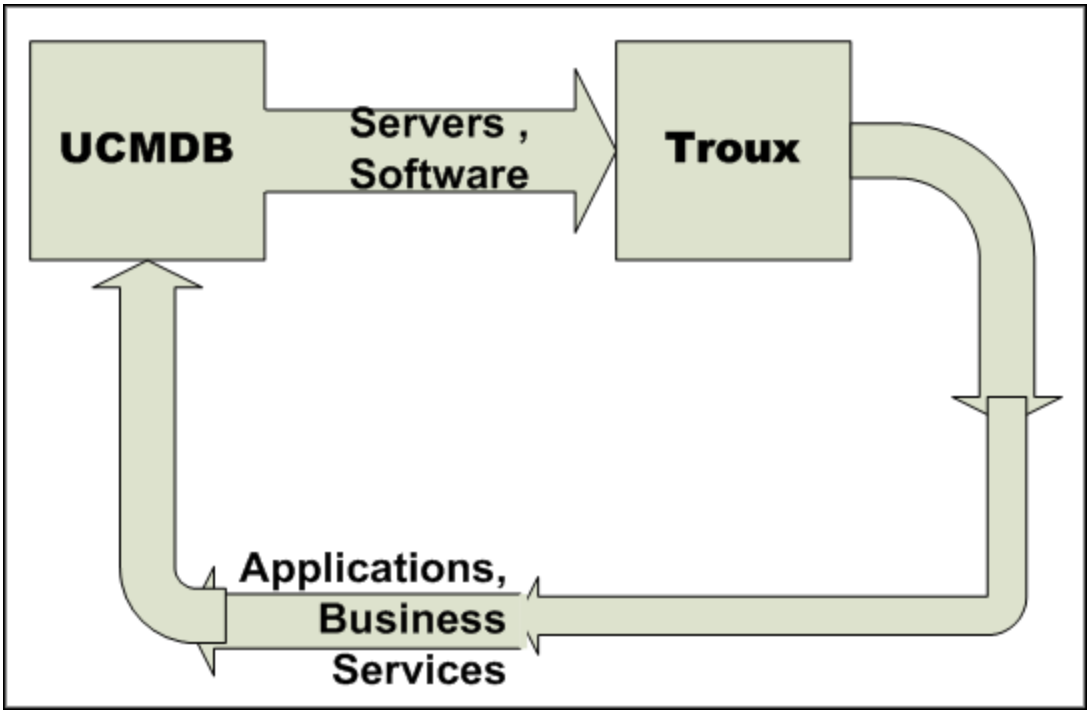
Although many use cases can be achieved using EA tools, two specific use cases were chosen for the UCMDB-Troux integration. This does not preclude additional use cases in the future. Depending on the use case, a provider of record is determined. For example, UCMDB would be the provider of record for inventory information such as the server operating system, server hardware, database, and other infrastructure CIs. Troux on the other hand provides component lifecycles for server operating system, server hardware, and database versions.

Integration Overview

UCMDB-Troux integration consists of two independent, bi-directional parts: the **Troux Push Adapter**, and the **Troux Population Adapter**.

- The Troux Push Adapter in UCMDB replicates CIs and relationships to Troux. The Troux Push Adapter is necessary to achieve both the Technology Standards Assessment and Business Impact Analysis use cases discussed in the introduction above. The adapter also allows the user to push to Troux CIs that are aged out of UCMDB or deleted.
- The Troux Population Adapter pulls CIs and relationships from Troux to UCMDB. It is necessary only for the Business Impact Analysis use-case.

Data transfer occurs using XML files between configured directories. Mapping files are used to apply conversion from TUX format to UCMDB and vice versa.



Supported Versions

Supported versions of the products are listed below.

Target Platform	DFM Protocol	UCMDB Version
Troux 9.x	None	9.02 and later

Use Cases

The use cases chosen for UCMDB-Troux integration are:

- **Technology Standards Assessment.** The ability to look at a Lifecycle of Software Products to determine viability within an enterprise.
- **Business Impact Analysis.** Definition of the definitive source of application CIs to align IT with business. These application CIs in Troux are related to server operating system, server hardware, database, and other CIs discovered by UCMDB. Impact Analysis can be determined using application, business function, and organization for planned change or unplanned disruption of service.

How to Work with the Troux Push Adapter

This adapter allows replication of CIs and links from UCMDB to Troux. This is accomplished by definition of queries and mapping files that define the CIs to be transferred and the naming/mapping of CIs and attributes to Troux components. This adapter also allows the user to push to Troux CIs that are aged out of UCMDB or deleted.

This task includes:

Define queries

1. Create a query that defines the CIs and attributes you want to replicate to Troux. Two example queries are supplied in the **Integration > Troux** folder.

   For details, see "Topology Query Language" in the *HP Universal CMDB Modeling Guide*.

2. Define the properties of each of the CITs.

   Note: This step is critical to the operation of the push adapter. You must define the

attributes that will be transferred to Troux.

For details, see "Query Node/Relationship Properties Dialog Box" in the *HP Universal CMDB Modeling Guide*.

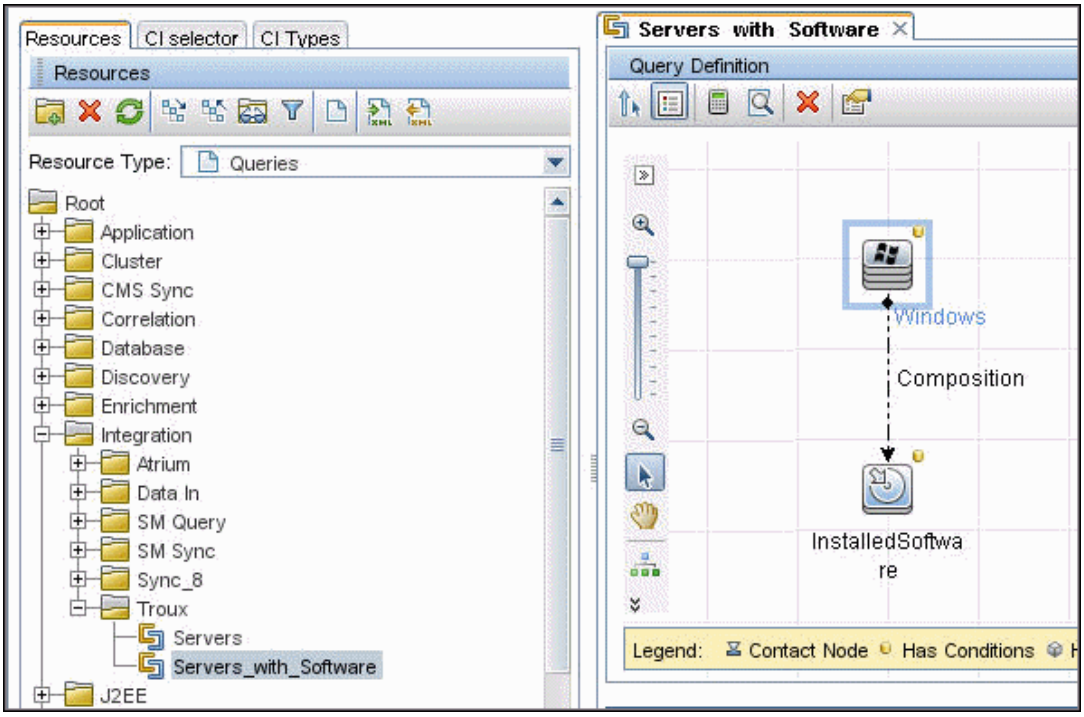
a. Define the criteria for the Query Node properties

b. Define the advanced properties for each of the attributes.

c. Select attributes to be transferred to Troux.

Example: Computers_for_Troux

In this example, the query requests UCMDB to send all computers with installed software to Troux. You must define the mapping file with the same name as the query in order for the push adapter to recognize the query.



Create mapping files

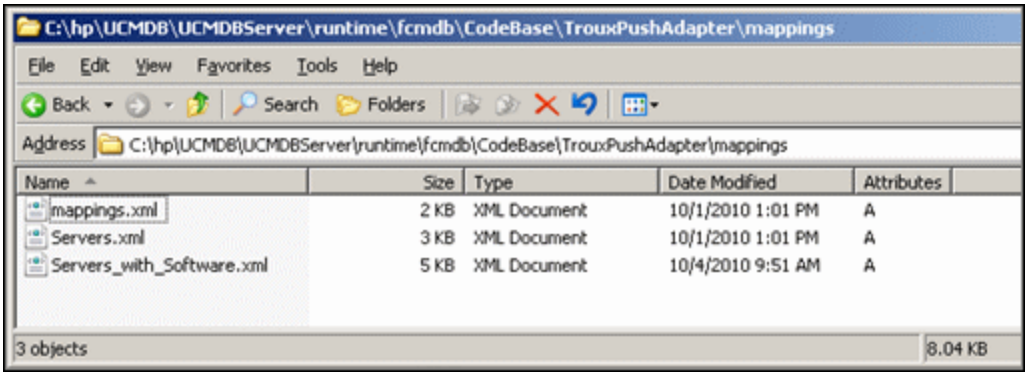
A mapping file is the translation template that defines the CITs and the relationships to be converted from UCMDB to Troux. For the push adapter to create output, this mapping file must have definitions for the attributes and CITs or relationships for export. The mapping file is located in: **`UCMDB\UCMDBServer\runtime\fcmdb\CodeBase\<adapter>\mappings`**

where `<adapter>` is the name of the adapter.

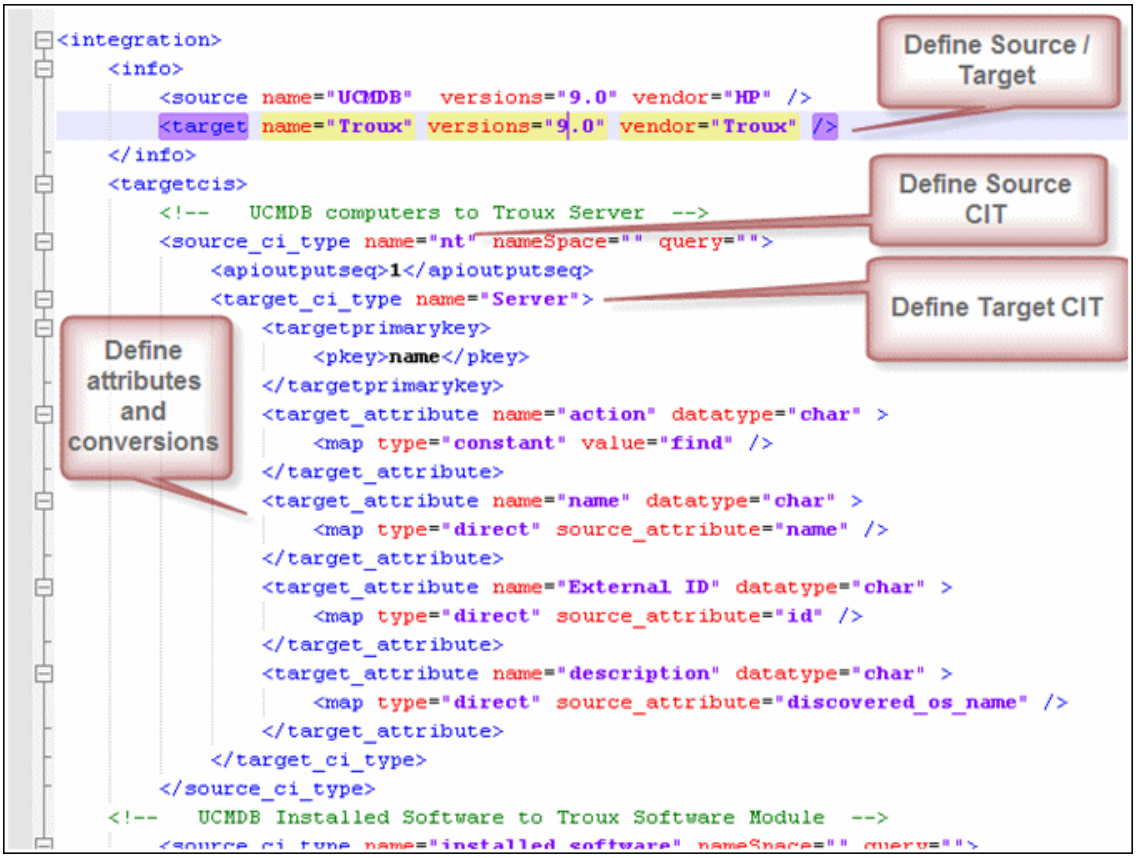
The example mapping file and query (Servers_with_Software) included with the content package sends Windows computers with installed software to Troux, as expected by Troux. If your environment uses different CIs with Troux, make sure Troux handles those component types.

When you create the mapping file, give it exactly the same name as your query. For details about the mapping file options, see "Prepare the Mapping Files" in the *HP Universal CMDB Developer Reference Guide*. Use the example mapping files as reference examples for the mapping file creation.

Note: The definitions in the mapping file (<adapter>.xml) must be the direct CITs and relationships to be transferred to Troux. The mapping does not support inheritance of class types. For example, if the query is transferring nt CITs, the mapping file must have definitions for nt CITs, and not for general nodes or computers. That is, the definition must be an exact match for what to transfer.



Example




```
<integration>
    <info>
        <source name="UCMDB"  versions="9.0" vendor="HP" />
        <target name="Troux" versions="9.0" vendor="Troux" />
    </info>
    <targetcis>
        <!--   UCMDB computers to Troux Server  -->
        <source_ci_type name="nt" nameSpace="" query="">
            <apioutputseq>1</apioutputseq>
            <target_ci_type name="Server">
                <targetprimarykey>
                    <pkey>name</pkey>
                </targetprimarykey>
                <target_attribute name="action" datatype="char" >
                    <map type="constant" value="find" />
                </target_attribute>
                <target_attribute name="name" datatype="char" >
                    <map type="direct" source_attribute="name" />
                </target_attribute>
                <target_attribute name="External ID" datatype="char" >
                    <map type="direct" source_attribute="id" />
                </target_attribute>
                <target_attribute name="description" datatype="char" >
                    <map type="direct" source_attribute="discovered_os_name" />
                </target_attribute>
            </target_ci_type>
        </source_ci_type>
    <!--   UCMDB Installed Software to Troux Software Module  -->
        <source_ci_type name="installed_software" nameSpace="" query="">
```

Create an integration point

This section describes how to create and run the job that replicates the data from UCMDB to Troux.

1. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
2. Click the **new integration point** button to open the new integration point Dialog Box.
   a. Click  , select the **Data Push into Troux** adapter and click **OK**.
   b. Enter the following information:

Name	Description
Integration Name	The name you give to the integration point.
Is Integration Activated	Select this check box to create an active integration point. You clear the check box if you want to deactivate an integration, for instance, to set up an integration point without actually connecting to a remote machine.
allowedComponentstodelete	The name of the component which is to be deleted and pushed to Troux. The default is **Server**. **Caution:** UCMDB is the system of record for servers. The default value of **Server** is important to note because the Troux system model may not allow deletion of other CITs. If you modify this property, you must verify (a) the mapping file is setup correctly to send the deleted CIT to Troux, and (b) Troux is setup to handle the delete for the CIT you specify. Deletion of **Server** is the only OOTB CIT that has been verified.
Probe Name	The name of the Data Flow Probe.
TUX path	The location of the TUX output file (created when the integration job is run).

   c. Click **Test connection** to verify the connectivity, and click **OK**. If the connection fails, verify the provided information is correct.

Define an integration job

You use the Integration tab to define a job that uses the integration point that you just defined. For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *HP Universal CMDB Data Flow Management Guide*.

1. Select the queries that you defined in "How to Work with the Troux Push Adapter" on page 721.
2. To enable deletion, select the **Allow Deletion** check box. This is a master delete **on/off** flag for the job that turns on and off the capability to send deletes.

   Note: Deleted CITs will be pushed depending on the changes to them in UCMDB. For example, if UCMDB discovers a computer, and that computer is deleted in UCMDB, this indicates a change of a deleted computer. And if this computer is part of the push query for Troux, with delete enabled, a delete is pushed to Troux the next time the push job is run.

3. Specify the job's schedule.
4. Click **OK**.
5. In the Integration Point pane, click **Save**. A full data push job will run according to schedule.

   The Troux output file (TUX) is generated in the path that you specified in the Integration Properties for the job.

Note: To run the job again, click  .

How to Run a Troux Population Job

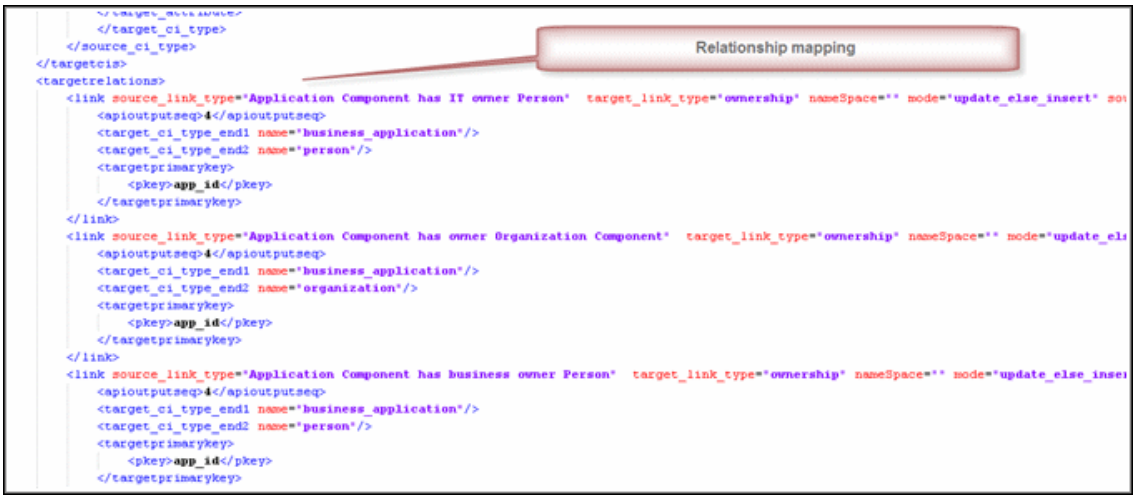
Prerequisite - Create a mapping file

Create a mapping file that enables mapping of Troux components and relationships to UCMDB CITs and relationships.

The top section of the file defines the object or CIT mapping from Troux to UCMDB. The lower section defines the relationship mapping.

The out-of-the-box mapping file, **Troux_to_UCMDB.xml** (located in **\DataFlowProbe\runtime\probeManager\discoveryResources\TQLExport\Troux\data**) contains the typical definitions for mapping the components and CITs with relationships.


```
<integration>
    <info>
        <source name="Troux"  versions="9.0" vendor="Troux" />
        <target name="UCMDB" versions="9.0" vendor="HP" />
    </info>
    <targetcis>
        <!-- Troux Application  -> UCMDB Application -->
        <source_ci_type name="Application" nameSpace="" query="">
            <apioutputseq>1</apioutputseq>
            <target_ci_type name="business_application">
                <targetprimarykey>
                    <pkey>name</pkey>
                </targetprimarykey>
                <target_attribute name="name" datatype="StrProp" >
                    <map type="direct" source_attribute="name" />
                </target_attribute>
                <target_attribute name="data_externalid" datatype="StrProp" >
                    <map type="direct" source_attribute="uuid" />
                </target_attribute>
                <target_attribute name="data_origin" datatype="StrProp" >
                    <map type="direct" source_attribute="extalias" />
                </target_attribute>
            </target_ci_type>
        </source_ci_type>
        <!-- Troux Person  -> UCMDB Person -->
        <source_ci_type name="Person" nameSpace="" query="">
            <apioutputseq>1</apioutputseq>
            <target_ci_type name="person">
                <targetprimarykey>
                    <pkey>given_name</pkey>
                </targetprimarykey>
                <target_attribute name="name" datatype="StrProp" >
                    <map type="direct" source_attribute="name" />
                </target_attribute>
```



Run the job - UCMDB 9.04 and later

Note: For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

In DFM, in the Integration Studio, create a new integration point.

1. Provide a name and description for the integration point.
2. Under **Integration Properties > Adapter**, select the **Population from Troux** adapter.
3. Edit the **TUX path** field, if required; this sets the location of the TUX output file.
4. Under **Adapter Properties > Probe Name** select the **DataFlow Probe**.
5. Under **Adapter Properties > Trigger CI instance** select:
   a. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or

   b. **Create New CI** (if you need to create a new CI). The **Topology CI Creation** Wizard appears. Complete the creation of the CI using the Wizard.

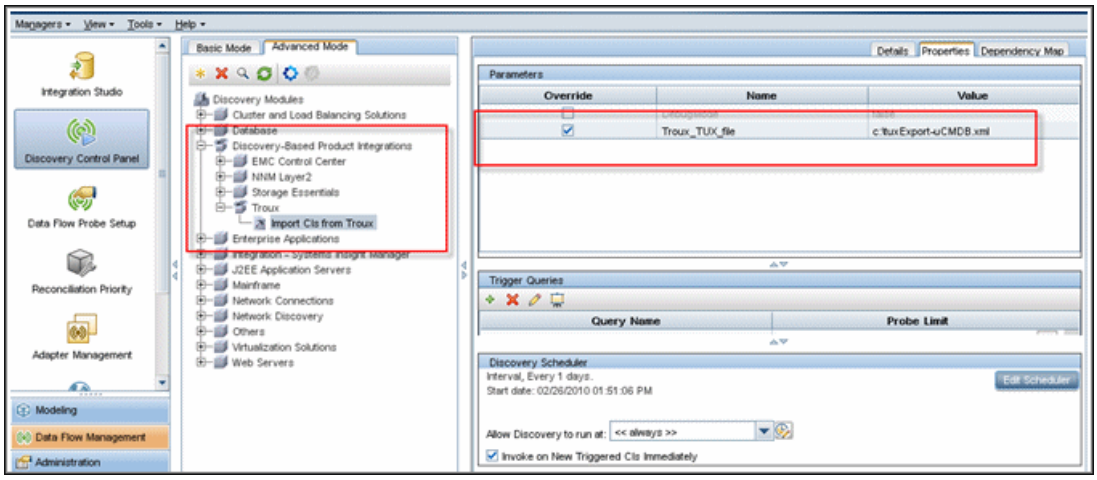
   Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *HP Universal CMDB Data Flow Management Guide*.
6. Save the integration point.
7. Run the job.

Note: The remaining steps are for UCMDB 9.03 and 9.02 only.

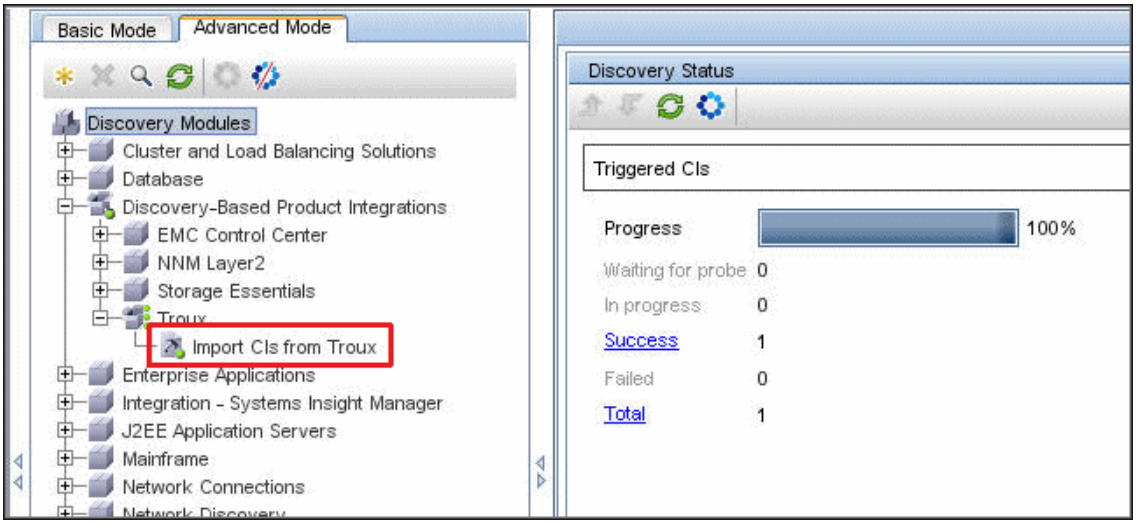
Activate the import job - UCMDB 9.03 and 9.02

1. In UCMDB, navigate to the Discovery Control Panel.
2. Open the **Discovery-Based-Product-Integrations** > **Troux** folder.

3. Select the **Import CIs from Troux** job.
4. In the **Properties** tab, replace **Troux_TUX_file** with the location of the TUX file that was output by Troux to import into UCMDB.



Activate the job - UCMDB 9.03 and 9.02



You can see the running of the job in the WrapperProbe log.

Chapter 57

UCMDB to XML Adapter

Note: This functionality requires UCMDB 9.04 Cumulative Update Package (CUP) 3.

This chapter includes:

Overview

By using the UCMDB to XML adapter, it is possible to export the results (CIs and relationships) of TQL queries and convert these to XML files.

Integration Mechanism

After defining an integration point with the UCMDB to XML adapter, TQL queries can be added to the jobs of that integration point.

The adapter exports the result of the TQL queries into XML format, and creates XML files in the Export Directory (as predefined in the integration point).

How to Export UCMDB to XML

1. **Prerequisite - General**
   a. Create a directory on the UCMDB data flow probe system to which the adapter will write the exported XML files.
   b. In the Modeling Studio, create integration TQL queries for the data to be exported to XML. Ensure the queries return valid results.
2. **Prerequisite - Create an integration point and integration job**

   In DFM, in the Integration Studio, create a new integration point.
   a. Provide a name and description for the integration point.
   b. Under **Integration Properties > Adapter**, select **UCMDB to XML** adapter.
   c. Under **Adapter Properties > Export Directory** type an absolute path of the directory on the probe system where the adapter should export the XML files to.
   d. Under **Adapter Properties > Probe Name**, select the name of the data probe to be used.
   e. Click the **Test Connection** button to ensure the adapter can validate the defined export directory.
   f. Save the integration point.
   g. Under **Integration Jobs** add a new integration job. Edit the job to add integration queries under the job's definition.

      For details on integration points, see **Integration Studio > Work with Data Push Jobs > Create an integration point** in the *HP Universal CMDB Data Flow Management Guide.*
   h. Click **OK** to save the integration job.
   i. Click the **Save** button to save the integration point.
3. **Run the job**
   a. To run the job ad hoc, select the integration job and click the  button. The job can also be configured to run on a schedule.

b. Check the defined export directory on the probe system for the exported XML data, to ensure the queries create valid XML files.

> **Note:** The XML files have time stamps in the format **YYMMDDHHMMSSZZZ**. If the integration query returns a large number of CIs, the export is by chunks of 1,000 CIs. Each chunk is a separate XML file, with the file for the last chunk having the string **EOD** (end of data) appended to it.

> **Note:** For details on running an integration job, see "Integration Studio" in the *HP Universal CMDB Data Flow Management Guide*.

Adapter

This job uses the adapter **UCMDB to XML** (XmlPushAdapter).

Used Scripts

pushToXml.py

Parameters

Parameter	Description
Export Directory	The absolute path (on the probe system) of the directory where the XML files will be exported to.
Probe Name	The name of the data flow probe to be used.

Part VII: Mainframe

Chapter 58

Mainframe by EView 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 743.

To run the discovery, see "How to Discover Mainframe by EView" on page 741.

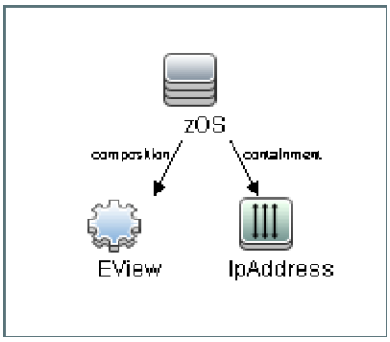
Supported Versions

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+

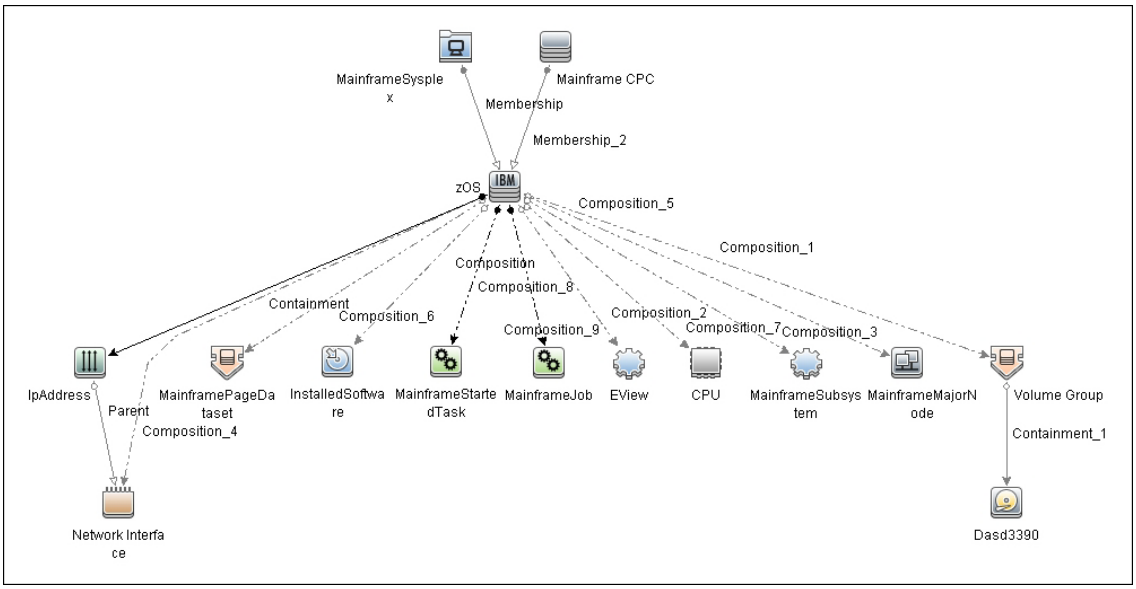
Topology

This section displays topology maps for the following jobs:

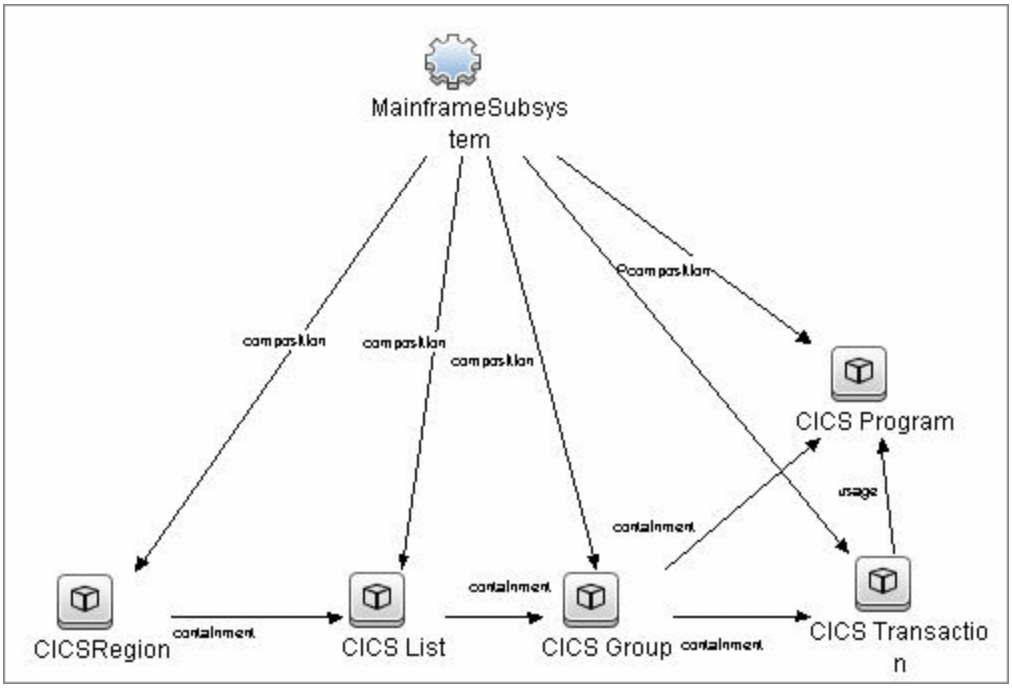
EView Connection



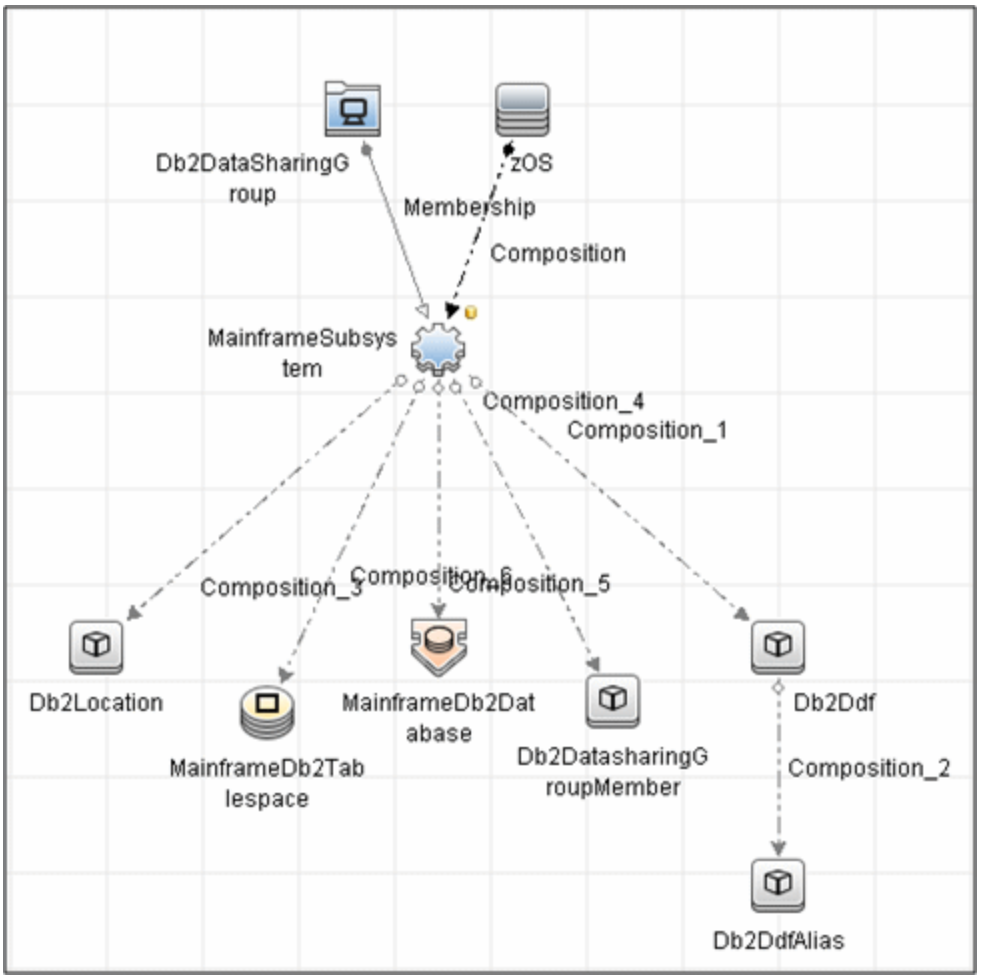
LPAR Resources by EView



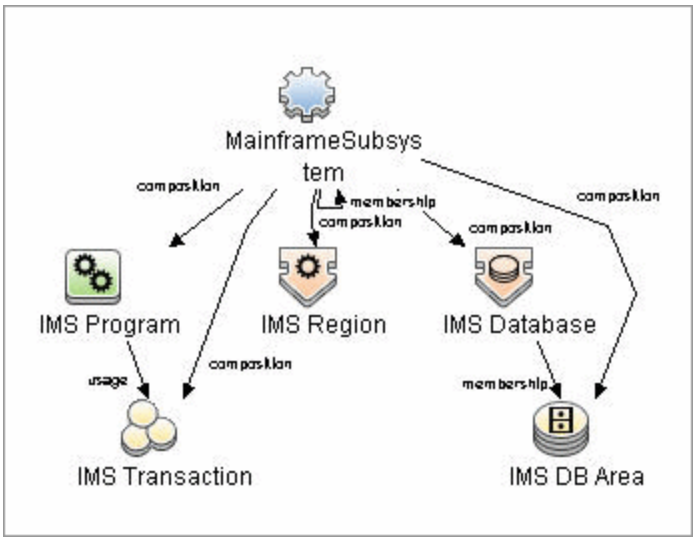
CICS by EView



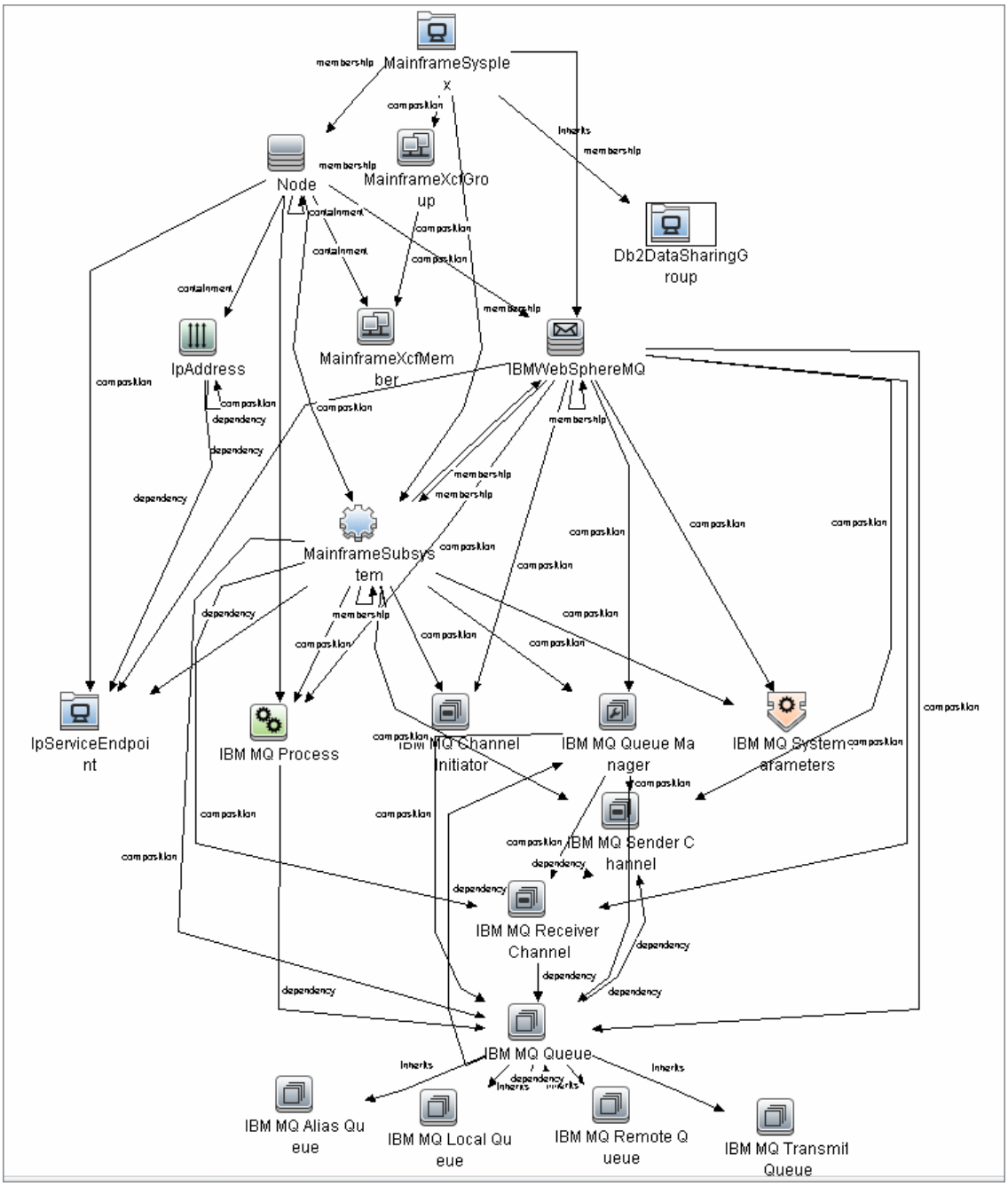
DB2 by EView


Db2DataSharingG
roup
zOS
Membership
Composition
MainframeSubsys
tem
Composition_4
Composition_1
Composition_3
Composition_6
Composition_5
Db2Location
MainframeDb2Tab
lespace
MainframeDb2Dat
abase
Db2DatasharingG
roupMember
Db2Ddf
Composition_2
Db2DdfAlias

IMS by EView


MainframeSubsys
tem
membership
IMS Program
IMS Region
IMS Database
usage
IMS Transaction
IMS DB Area

MQ by EView



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:

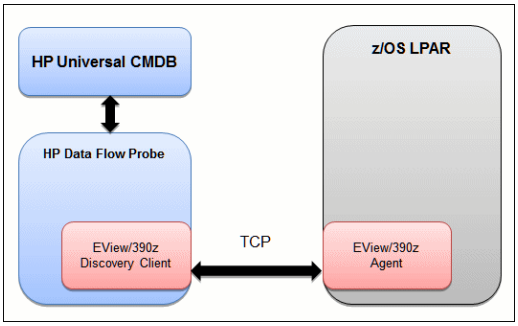
   - Activate the **LPAR Resources by EView** job to discover the z/OS LPAR host and network resources. For details about this job, see "LPAR Resources by EView Job" on page 745.
   - Activate the **CICS by EView** job to discover the CICS subsystem and its resources. For details about this job, see "CICS by EView Job" on page 747.
   - 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 748.
   - 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 749.
   - 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 750.

   For details on running jobs, refer to "Discovery Control Panel" 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 741.

EView Connection 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 737.

LPAR Resources by EView 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
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 738.

CICS by EView 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 738.

DB2 by EView 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 739.

IMS by EView 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 740.

MQ by EView 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 741.

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.

Chapter 58

Part VIII: Storage

Chapter 59

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.

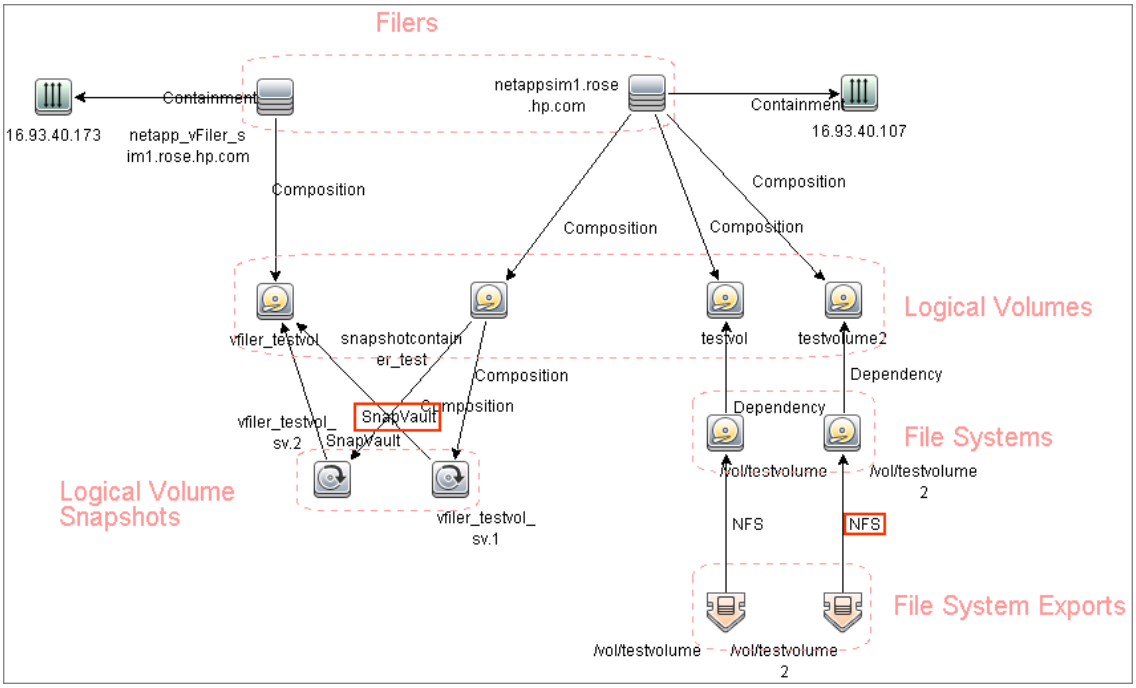
Supported Versions

This discovery supports NetApp Data ONTAP 7.2.x and 7.3.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 758



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" on page 82.

2. ## Prerequisite - Permissions

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

   Ensure the user has the appropriate permissions on the AS400 system to run the following discovery commands:

Command	Description
`system-get-info`	Get appliance details including CPU and backplane information. (Head information in a sysconfig -a command). I/O information is not included.
`system-get-ontapi-version`	Required to Get current ONTAPI major and minor versions.
`ipspace-list-info`	Get information about ipspaces including IP addresses and relevant IP details. (Requires **vfiler** license.)
`options-get`	Get values for optional parameters.

`volume-list-info-iter-start`
`volume-list-info-iter-next`
`volume-list-info-iter-end`
	Get details on volumes in the appliance.

`snapshot-list-info`	Get details on snapshots for a specified volume.

`snapvault-<SnapvaultLevel>`
`-relationship-status-list-iter-start`
`snapvault-<SnapvaultLevel>`
`-relationship-status-list-iter-next`
	Get snapvault details from the appliance. <SnapvaultLevel> can be either **primary** or **secondary** or both of these.

`cifs-share-list-iter-start`
`cifs-share-list-iter-next`
`cifs-share-list-iter-end`
	Get details on CIFS shares on this appliance. (Requires **cifs** license.)

	Get details on CIFS sessions on this

Command	Description

`cifs-session-list-iter-start`
`cifs-session-list-iter-next`
`cifs-session-list-iter-end`
	appliance. (Requires **cifs** license.)

`nfs-exports-list-rules`	Get details on NFS shares on this appliance.

3. **Run the discovery**

> **Note:** For details on running jobs, refer to "Discovery Control Panel" 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" below.

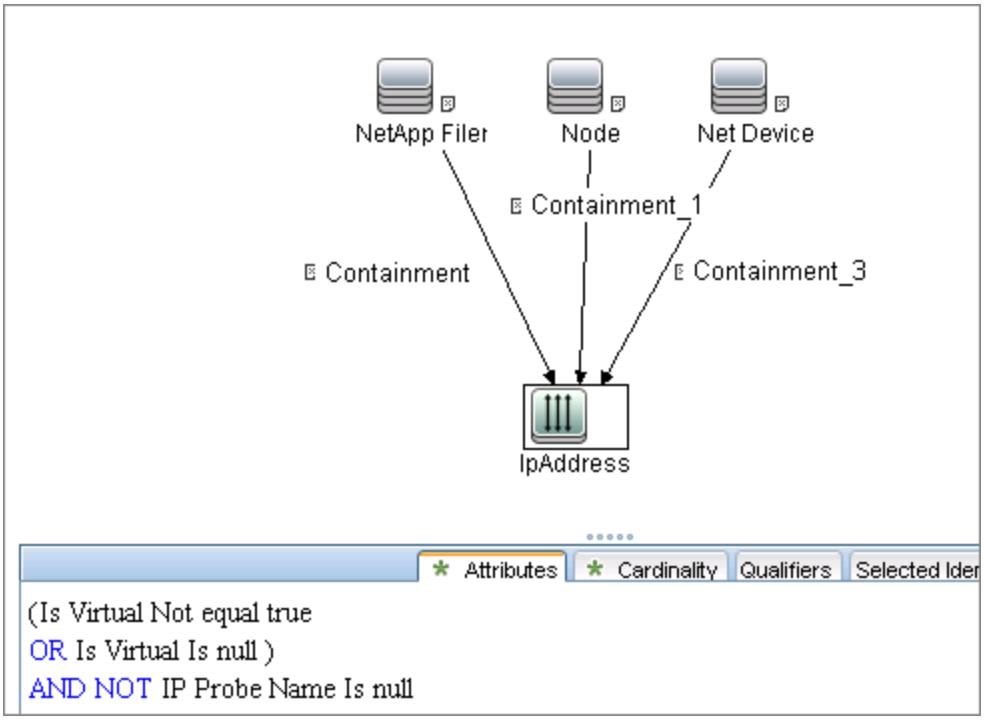
NetApp Filer by WebServices Job

The NetApp Filer discovery package is bundled in **NetAppFiler.zip**.

This section includes:

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).



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.
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.

Parameter	Description
	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

- **CPU**
- **Containment**
- **Dependency**
- **File System**
- **Node**
- **IpAddress**
- **Logical Volume**
- **Logical Volume Snapshot**
- **Membership**
- **Interface**
- **Realization**

Note: To view the topology, see "Topology" on page 754.

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.

Part IX: J2EE

Chapter 60

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 Resources and Applications by Shell** in order to discover applications of the target host, including running processes.
      - **J2EE TCP Ports** in order to discover service endpoint information.

3. Run the job **J2EE Glassfish by Shell**.

Glassfish_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:NA}

Used Scripts

- **glassfish_by_shell.py**
- **glassfish_discoverer.py**
- **glassfish.py**
- **process_discoverer.py**
- **jee_discoverer.py**
- **jms.py**
- **jee.py**
- **jee_connection.py**
- **connection.py**
- **jdbc.py**
- **jdbc_url_parser.py**
- **db.py**
- **db_builder.py**
- **db_platform.py**
- **entity.py**
- **jmx.py**
- **iteratortools.py**

- **protocol.py**
- **jdbcutils.py**
- **j2eeutils.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

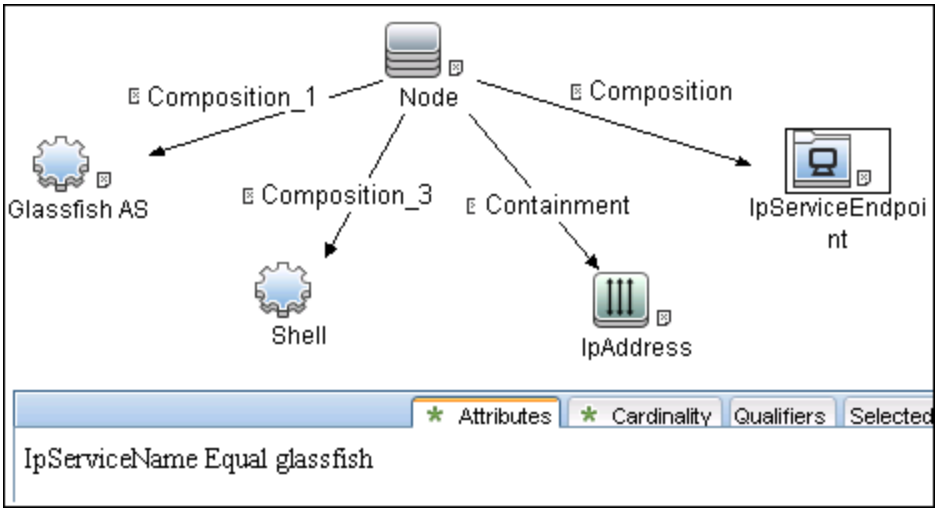
- **globalSettings.xml**

Parameters

- **reportAdminApp**s - enables/disables reporting of administrator applications if value is 'true'/'false'

Glassfish_By_Shell Job

Trigger Query



Parameters

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 "glassfish"

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 **clearProbeData.bat** script
  - Start the probe

Chapter 61

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 J2EE 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 770

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

   This discovery uses the JBoss protocol.

   For credential information, see "Supported Protocols" on page 82.

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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

- Run the **Range IPs by ICMP** job to discover the target IPs.
- Run the **J2EE TCP Ports** job to discover service endpoint information. For job details, see "J2EE TCP Ports Job" on next page"J2EE TCP Ports Job" on next page"J2EE TCP Ports Job" on next page.
- Run the **J2EE JBoss Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "J2EE JBoss Connections by JMX Job" on page 776.
- Run the **J2EE JBoss by JMX** job to perform a deep discovery of JBoss application server topology. For job details, see "J2EE JBoss by JMX Job" on page 779.

How to Discover J2EE 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" on page 82.

   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 "Discovery Control Panel" 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 Resources and Applications by Shell** to discover applications of the target host, including running processes.
      - **J2EE TCP Ports** to discover service endpoint information. For job details, see "J2EE TCP Ports Job" below.

   d. Run the **J2EE JBoss by Shell** job. For job details, see "J2EE JBoss by Shell Job" on page 782.

J2EE TCP Ports Job

This section includes:

Trigger Query

Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

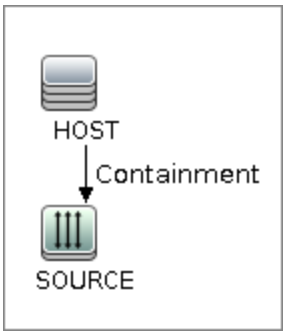
Job Parameters

Name	Value	Description
ports	weblogic,weblogicSSL,websphere,rmi	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known ports or not according to the checkOnlyKnownPorts parameter

Adapter - TCP_NET_Dis_Port

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**
- **Global Configuration File**
  - **portNumberToPortName.xml**
- **Parameters**

Name	Value	Description
checkIfIpIsReachable	true	Flag that indicates whether to check if the discovered IP is reachable before its ports are pinged (true/false).
checkOnlyKnownPorts	true	Discover only known ports. This flag does not cancel the 'ports' parameter - overriding this flag to false is applicable only with real ports range in the 'ports' parameter.
connectTimeOut	5000	The timeout when connecting to IP and port.
pingTimeOut	2000	ICMP ping timeout (in milliseconds).
ports	*	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known

Name	Value	Description
		ports or not according to the checkOnlyKnownPorts parameter.

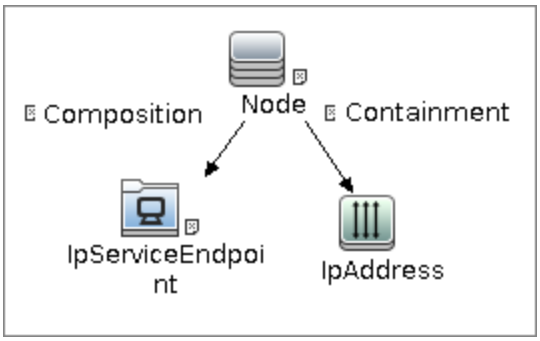
Discovered CITs

- **Composition**
- **Containment**
- **IpAddress**
- **IpServiceEndpoint**
- **Node**

J2EE JBoss Connections by JMX Job

This section includes:

Trigger Query



- **Node Conditions**

Node Name	Condition
Node	None
IpServiceEndPoint	`IpServiceName Equal "rmi"`
IpAddress	`NOT IP Probe Name Is null`

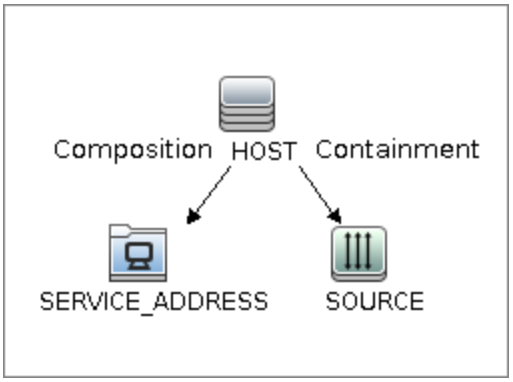
Job Parameters

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

Adapter - JMX_J2EE_JBoss_Connection

This adapter discovers JBoss servers 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**

- **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.

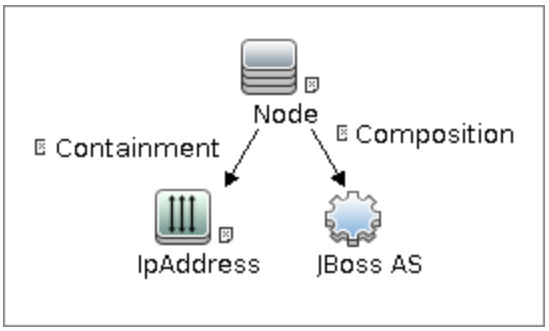
Discovered CITs

- **Composition**
- **Containment**
- **IpAddress**
- **IpServiceEndPoint**
- **J2EE Domain**
- **JBoss AS**
- **JEE Node**
- **Membership**
- **Node**
- **Usage**

J2EE JBoss by JMX Job

This section includes:

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`

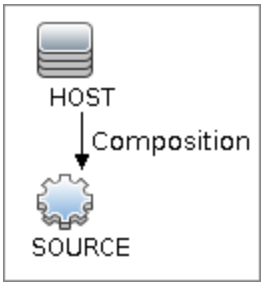
Job Parameters

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	Should pattern run in separate thread.

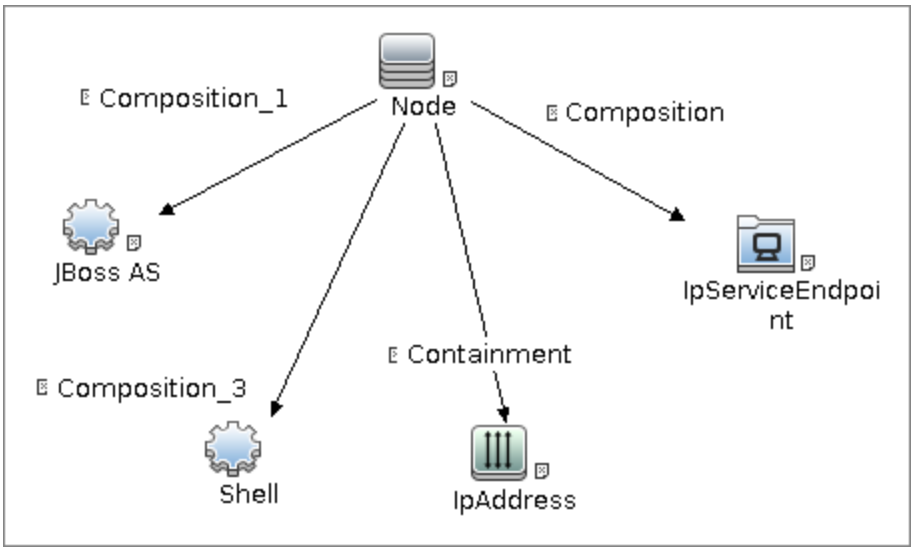
Discovered CITs

- **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**

J2EE JBoss by Shell Job

This section includes:

Trigger Query



- **Node Conditions**

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"

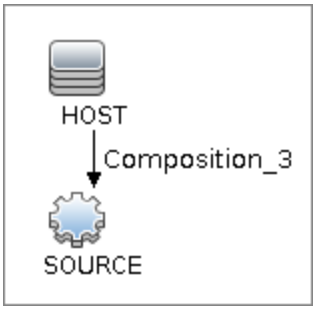
Job Parameters

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

Adapter - JBoss_By_Shell

- **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:NA}

- **Used Scripts**
  - **connection.py**
  - **db.py**
  - **db_builder.py**
  - **db_platform.py**
  - **entity.py**
  - **file_ver_lib.py**
  - **iteratortools.py**
  - **j2eeutils.py**
  - **jboss.py**
  - **jboss_by_shell.py**
  - **jboss_discoverer.py**
  - **jdbc.py**
  - **jdbc_ulr_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 62

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.

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 next page

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" on page 82.

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 "Discovery Control Panel" 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 **J2EE TCP Ports** job to discover service endpoint information. For job details, see "J2EE TCP Ports Job" on page 792.

c. Run the **J2EE Weblogic Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "J2EE Weblogic Connections by JMX Job" on

page 795.

d. Run the **J2EE Weblogic by JMX** job to perform a deep discovery of application server topology. For job details, see "J2EE Weblogic by JMX Job" on page 798.

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" on page 82.

2. **Discovery Workflow**

   For details on running jobs, refer to "Discovery Control Panel" 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 Resources and Applications by Shell** to discover resources of the target host, including running processes.
      - **J2EE TCP Ports** to discover service endpoint information. For job details, see "J2EE TCP Ports Job" on next page.

   d. Run the job **J2EE Weblogic by Shell**. For job details, see "J2EE Weblogic by Shell Job" on page 802.

J2EE TCP Ports Job

This section includes:

Trigger Query

- Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

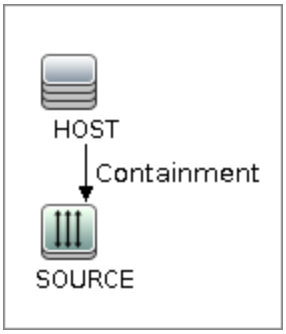
Job Parameters

Name	Value	Description
ports	weblogic,weblogicSSL,websphere,rmi	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known ports or not according to the checkOnlyKnownPorts parameter

Adapter - TCP_NET_Dis_Port

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**
- **Global Configuration File:** portNumberToPortName.xml
- **Parameters**

Name	Value	Description
checkIfIpIsReachable	true	Flag that indicates whether to check if the discovered IP is reachable before its ports are pinged (true/false).
checkOnlyKnownPorts	true	Discover only known ports. This flag does not cancel the 'ports' parameter - overriding this flag to false is applicable only with real ports range in the 'ports' parameter.
connectTimeOut	5000	The timeout when connecting to IP and port.
pingTimeOut	2000	ICMP ping timeout (in milliseconds).
ports	*	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known ports or not according to the checkOnlyKnownPorts parameter.

Discovered CITs

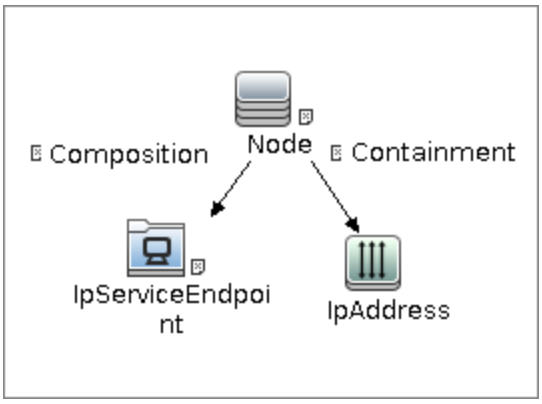
- **Composition**
- **Containment**
- **IpAddress**

- **IpServiceEndpoint**
- **Node**

J2EE Weblogic Connections by JMX Job

This section includes:

Trigger Query



- **Node Conditions**

Node Name	Condition
Node	None
IpServiceEndPoint	`IpServiceName Equal "weblogic"`
IpAddress	`NOT IP Probe Name Is null`

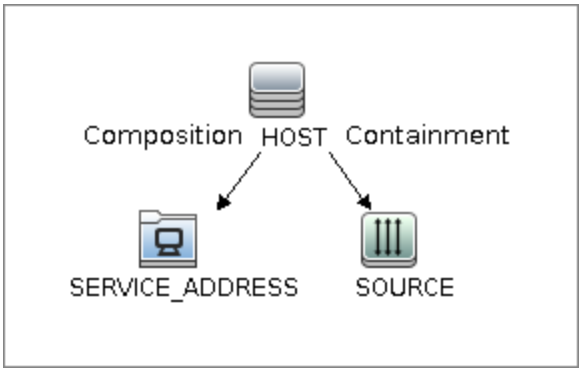
Job Parameters

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.

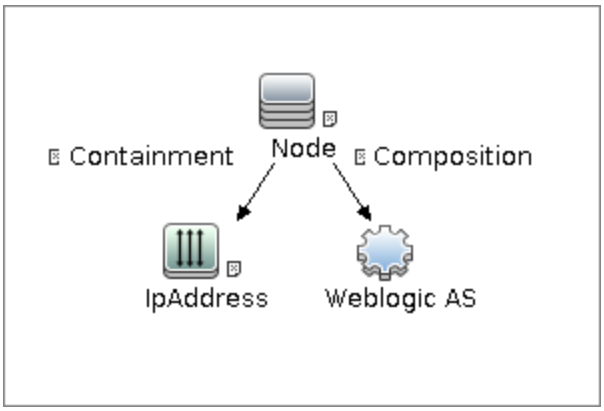
Discovered CITs

- **Composition**
- **Containment**
- **IpAddress**
- **IpServiceEndPoint**
- **J2EE Domain**
- **JEE Node**
- **JVM**
- **Membership**
- **Node**
- **Usage**
- **WebLogic AS**

J2EE Weblogic by JMX Job

This section includes:

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`

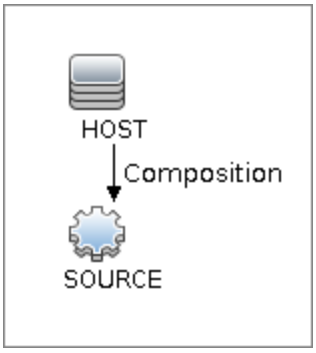
Job Parameters

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

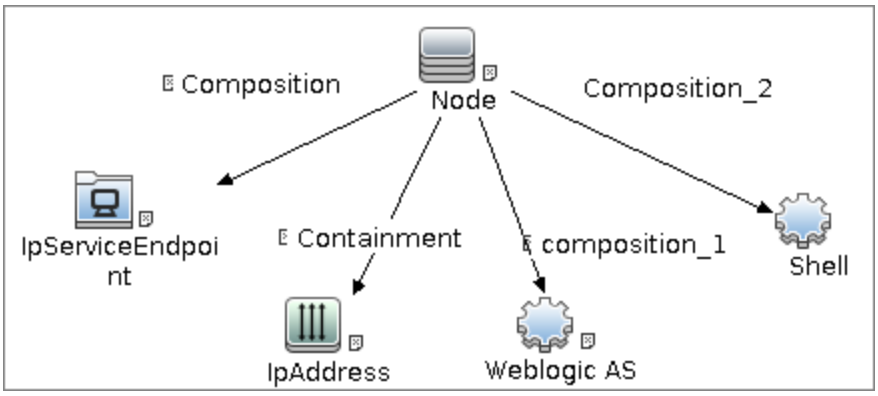
- **Composition**
- **Containment**
- **Dependency**
- **Deployed**
- **Membership**
- **Usage**
- **ConfigurationDocument**
- **Weblogic AS**
- **Database**
- **IpAddress**

- **IpServiceEndPoint**
- **J2EE Domain**
- **J2EE Cluster**
- **J2EE Managed Object**
- **JDBC Data Source**
- **JEE Node**
- **Node**
- **Web Service**
- **J2EE Execute Queue**

J2EE Weblogic by Shell Job

This section includes:

Trigger Query



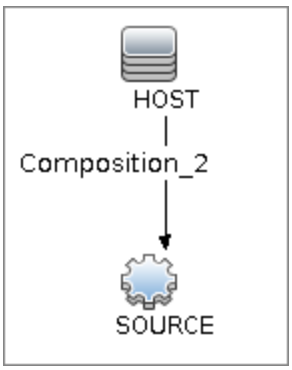
Job Parameters

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

Adapter - WebLogic_By_Shell

This adapter is used for Weblogic J2EE Topology Discovery by Shell.

- **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: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
  - 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**

- **Dependency**
- **Deployed**
- **Membership**
- **Usage**
- **ConfigurationDocument**
- **Weblogic AS**
- **Database**
- **IpAddress**
- **IpServiceEndPoint**
- **J2EE Domain**
- **J2EE Cluster**
- **J2EE Managed Object**
- **JDBC Data Source**
- **Node**
- **Web Service**

Troubleshooting and Limitations

- **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 (J2EE WebLogic Connections by JMX job and J2EE WebLogic by JMX job) by adding the following argument:

  ```
  Djavax.net.ssl.trustStore=..\runtime\probeManager\discoveryResources
  \j2ee\websphere\UCMDB_store.jks
  ```

Limitations

- WebLogic servers cannot be discovered if the WebLogic domain is configured with a domain-wide administration port. To enable discovery, access the WebLogic administrator console. In the Domain pane, clear the **Enable Administration Port** check box and save the changes.
- DFM discovers domains only when they are created by the WebLogic Configuration Wizard.
- For versions earlier than WebLogic 9, the J2EE 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 63

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" on page 82.

2. **Prerequisite - Set up drivers**

   Set up the drivers needed to discover WebSphere. Default WebSphere drivers are included by default with the Probe installation. For details on the required *.jar files, see "WebSphere" in the *HP Universal CMDB Data Flow Management Guide*.

   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. **Run the discovery**

   Run the following jobs in the following order:

   For details on running jobs, refer to "Discovery Control Panel" 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 **J2EE TCP Ports** job to discover service endpoint information. For job details, see "J2EE TCP Ports Job" on page 812.

   c. Run the **J2EE WebSphere Connections by JMX** job to perform a shallow discovery of application servers. For job details, see "J2EE WebSphere Connections by JMX Job" on page 815.

d. Run the **J2EE WebSphere by JMX** job to perform a deep discovery of application server topology. For job details, see "J2EE Websphere by Shell or JMX Job" on page 818.

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" below

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" on page 82.

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 "Discovery Control Panel" 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 Resources and Applications by Shell** job to discover applications of the target host, including running processes.
- Run the **J2EE TCP Ports** job to discover service endpoint information. For job details, see "J2EE TCP Ports Job" on next page.

d. Run the **J2EE WebSphere by Shell** job. For job details, see "J2EE Websphere by Shell Job" on page 822.

J2EE TCP Ports Job

This section includes:

Trigger Query

- Node Conditions

Node Name	Condition
IpAddress	`NOT IP Probe Name Is null`

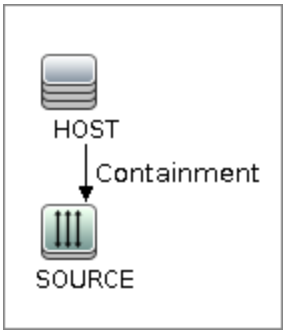
Job Parameters

Name	Value	Description
ports	weblogic,weblogicSSL,websphere,rmi	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known ports or not according to the checkOnlyKnownPorts parameter

Adapter - TCP_NET_Dis_Port

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**
- **Global Configuration File:** portNumberToPortName.xml
- **Parameters**

Name	Value	Description
checkIfIpIsReachable	true	Flag that indicates whether to check if the discovered IP is reachable before its ports are pinged (true/false).
checkOnlyKnownPorts	true	Discover only known ports. This flag does not cancel the 'ports' parameter - overriding this flag to false is applicable only with real ports range in the 'ports' parameter.
connectTimeOut	5000	The timeout when connecting to IP and port.
pingTimeOut	2000	ICMP ping timeout (in milliseconds).
ports	*	List of ports, can include ranges, separate port numbers and known protocol names (like http, ftp, etc) comma separated. Empty or * : all known ports. Also accepts ranges like 1000 - 1100 which would be filtered to known ports or not according to the checkOnlyKnownPorts parameter.

Discovered CITs

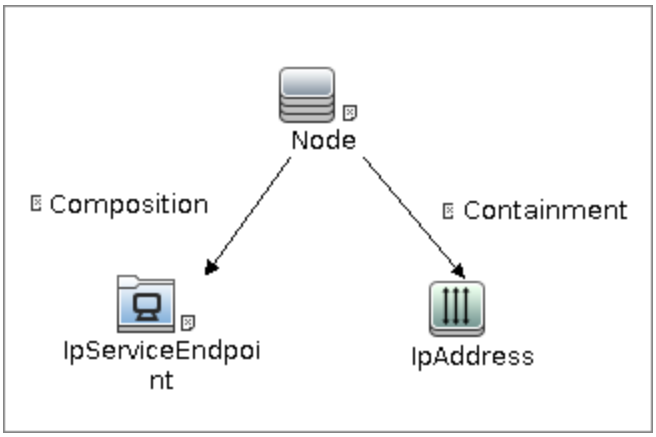
- **Composition**
- **Containment**
- **IpAddress**

- **IpServiceEndpoint**
- **Node**

J2EE WebSphere Connections by JMX Job

This section includes:

Trigger Query



Node Conditions

Node Name	Condition
Node	None
IpServiceEndPoint	`IpServiceName Equal "websphere"`
IpAddress	`NOT IP Probe Name Is null`

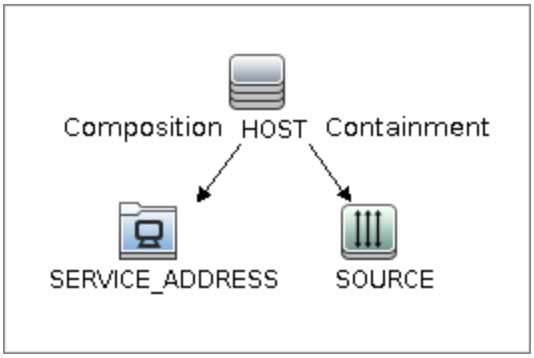
Job Parameters

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.

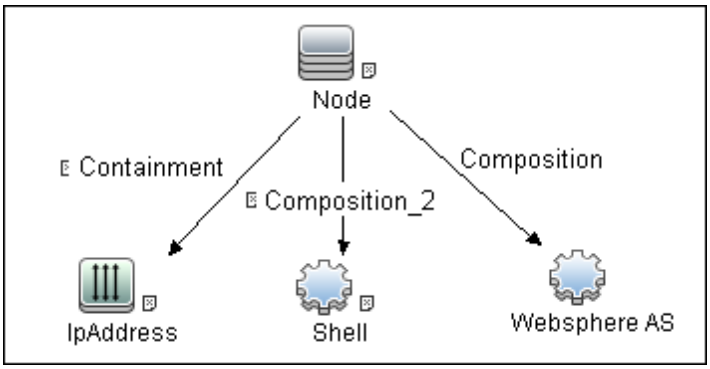
Discovered CITs

- **Composition**
- **IpAddress**
- **IpServiceEndPoint**
- **J2EE Domain**
- **JEE Node**
- **Node**
- **Usage**
- **Websphere AS**

J2EE Websphere by Shell or JMX Job

This section includes:

Trigger Query



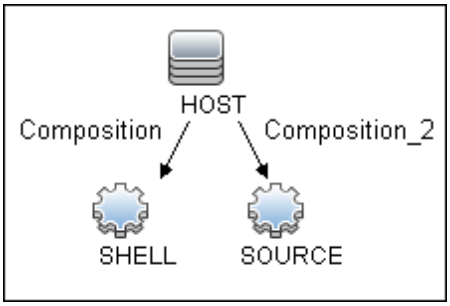
Job Parameters

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

Adapter - JMX_J2EE_WebSphere

This adapter is used for WebSphere J2EE Topology Discovery by JMX.

- **Input CIT:** WebSphere AS
- **Input Query**



- **Triggered CI Data**

Name	Value
credentialsId	`${SOURCE.credentials_id}`
hostId	`${HOST.root_id}`

Name	Value
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.
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	Should pattern run in separate thread.
servers	None	List of servers to discover (comma separated).

Discovered CITs

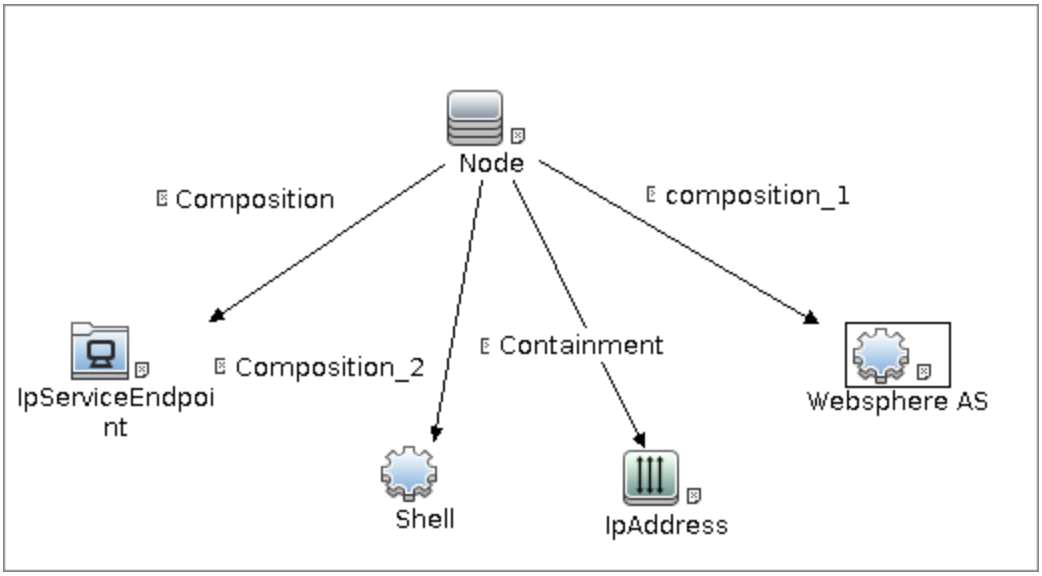
- **Composition**
- **ConfigurationDocument**
- **Database**
- **Dependency**
- **Deployed**
- **IpAddress**
- **IpServiceEndPoint**
- **J2EE Cluster**
- **J2EE Domain**
- **J2EE Managed Object**
- **JDBC Data Source**
- **JEE Node**

- **Membership**
- **Node**
- **Usage**
- **Web Service**
- **Websphere AS**

J2EE Websphere by Shell Job

This section includes:

Trigger Query



- **Node Conditions**

Node Name	Condition
Node	None
IpAddress	`NOT IP Probe Name Is null`
Websphere AS	NOT Reference to the credentials dictionary entry Is null

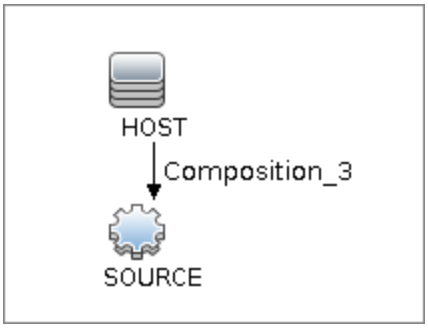
Job Parameters

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

Adapter - WebSphere_By_Shell

This adapter is used for Websphere J2EE Topology Discovery by Shell.

- **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:NA}

- **Used Scripts**
  - **connection.py**
  - **core.py**
  - **db.py**
  - **db_builder.py**
  - **db_platform.py**
  - **entity.py**
  - **file_ver_lib.py**
  - **iteratortools.py**
  - **j2eeutils.py**
  - **jdbcutils.py**
  - **jee.py**
  - **jee_connection.py**
  - **jee_discoverer.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.

- **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 (J2EE WebSphere Connections by JMX job and J2EE 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 X: Network

Chapter 64

Active and Passive Network Connections Discovery

This chapter includes:

Overview

All jobs in these modules run queries against the Data Flow Probe's MySQL 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 872.

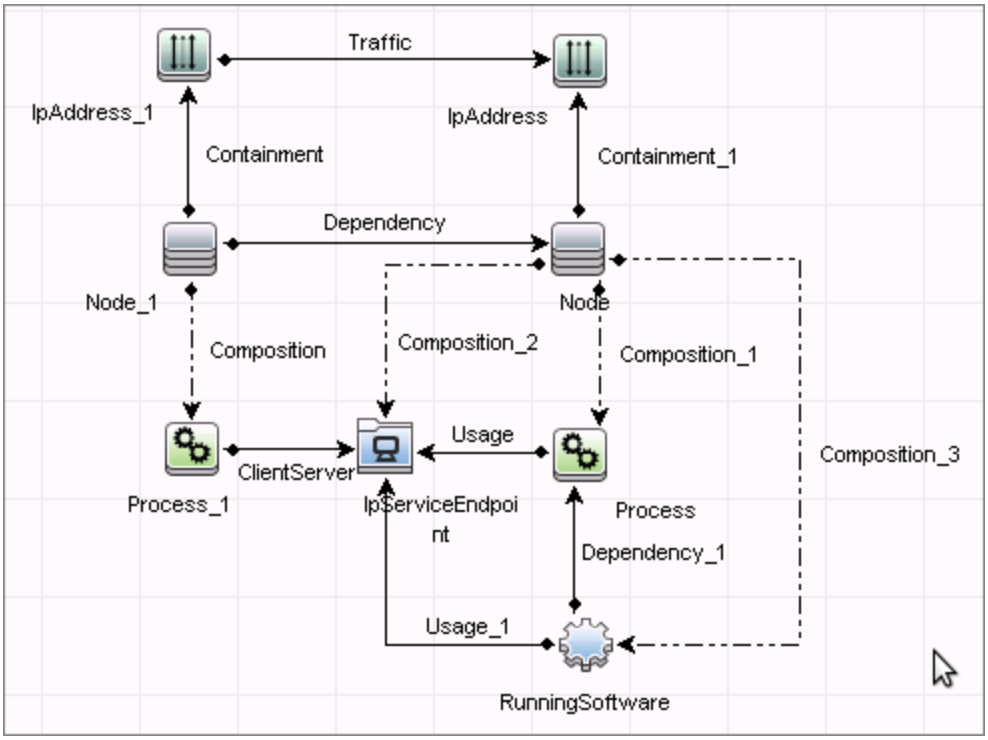
The Data Flow Probe includes a built-in MySQL database so there is no need to install a separate MySQL instance for NetFlow. Instead, data is saved to a dedicated scheme (called `netflow` for historical reasons).

Supported Versions

DFM supports NetFlow versions 5 and 7.

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" on page 82.

   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 MySQL database with TCP information gathered from the remote machine. For details, see "TCP Traffic Jobs" below.
   - Run the **Network Connectivity Data Analyzer** job. For job details, see "Network Connectivity Data Analyzer Job" on next page.

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 see "Host Resources and Applications Discovery" on page 872.
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 834.

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 '*', 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 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. There are two kinds of filters: Range filters and Service filters

Note: A host is filtered if at least one of the filters is applied to it.

- 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:

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.

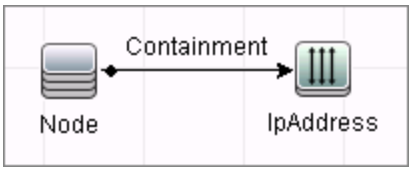
Note: A reporter can be deactivated if its active attribute is set to **false** or the tag responsible for the reporter is commented out or removed.

The following reporters are available:

- **Default**. For details, see "Default Reporter" below.
- **clientProcess**. For details, see "Client Process Reporter" on next page.
- **clientServerLink.** For details, see "Client Server Link Reporter" on page 839.
- **ipTrafficLink.** For details, see "IP Traffic Link Reporter" on page 840.
- **nodeDependencyLink.** For details, see "Node Dependency Link Reporter" on page 841.
- **serverProcess.** For details, see "Server Process Reporter" on page 842.
- **serverRunningSoftware.** For details, see "Server Running Software Reporter" on page 843.

Default Reporter

If no reporters are activated, the job returns the **IP** and **Node** CIs linked by the **containment** relationship only.



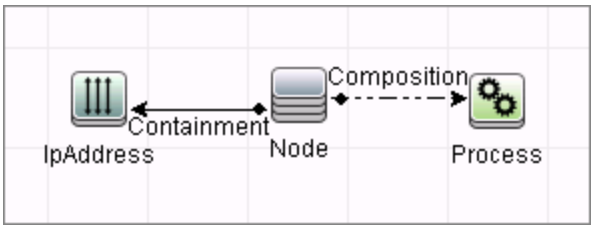
- ## Client Process Reporter

This reporter reports client processes.

reporter definition

```
<reporting>
    <reporter name="clientProcess" active="true"/>
</reporting>
```

Topology



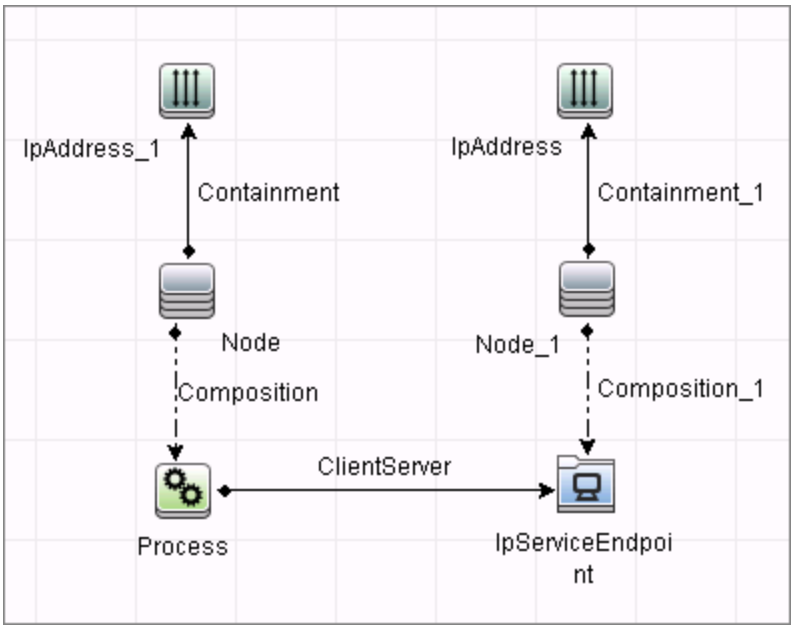
- ## Client Server Link Reporter

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



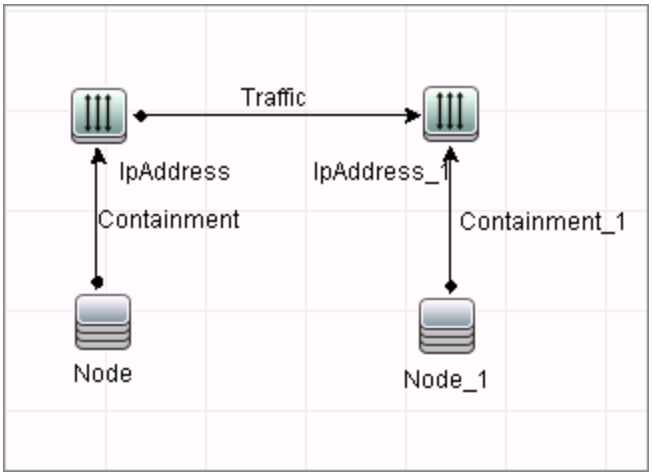
- ## IP Traffic Link Reporter

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



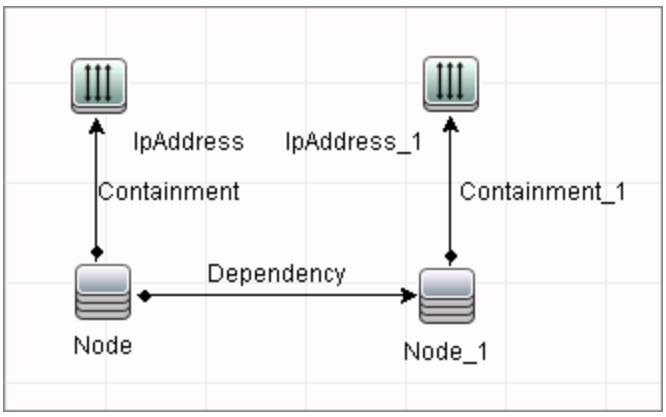
- ## Node Dependency Link Reporter

This reporter reports the dependency link between discovered nodes.

reporter definition

```
<reporting>
    <reporter name="nodeDependencyLink" active="true"/>
</reporting>
```

Topology



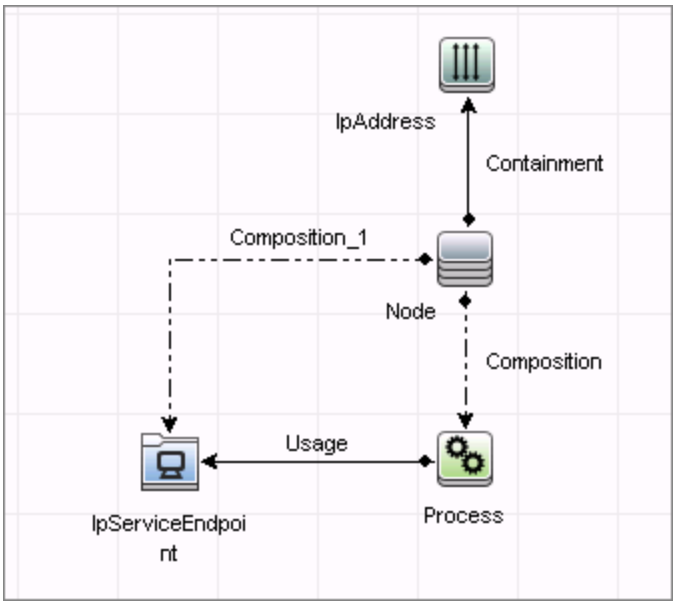
- ## Server Process Reporter

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



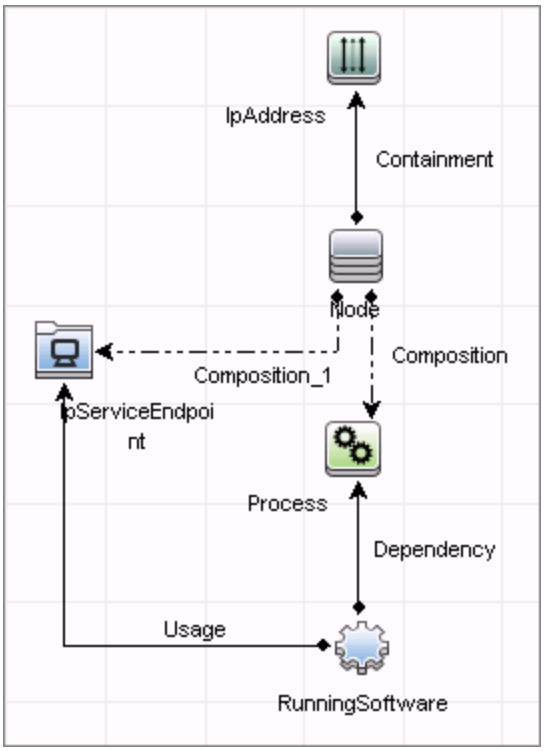
- ## Server Running Software Reporter

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). Server running software is reported only if the service it is representing is defined as **discover="1"** in the **portNumberToPortName.xml** file.

reporter definition

```
<reporting>
    <reporter name="serverRunningSoftware" active="true" linkWithProcess="true"/>
</reporting>
```

Topology



Chapter 65

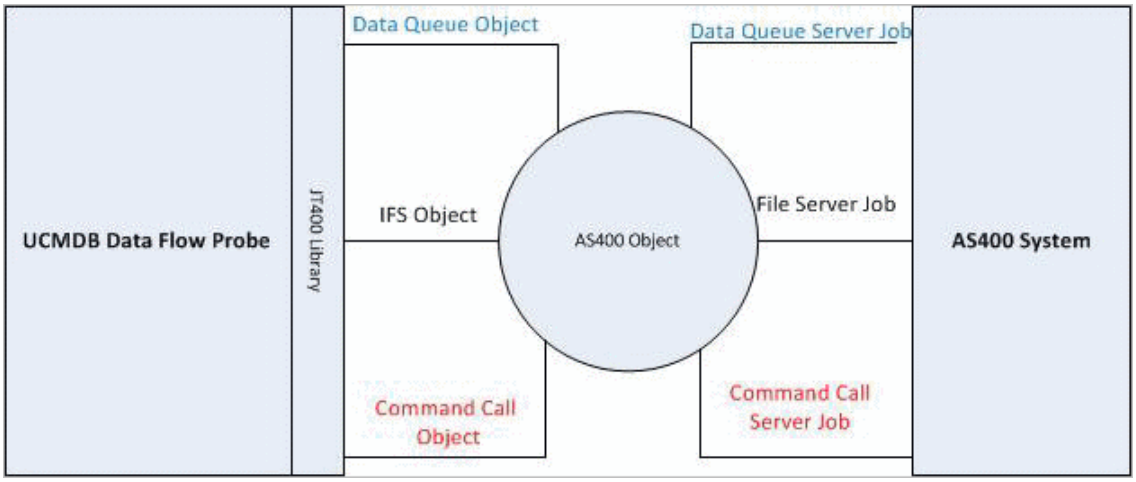
AS400 Host Discovery

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

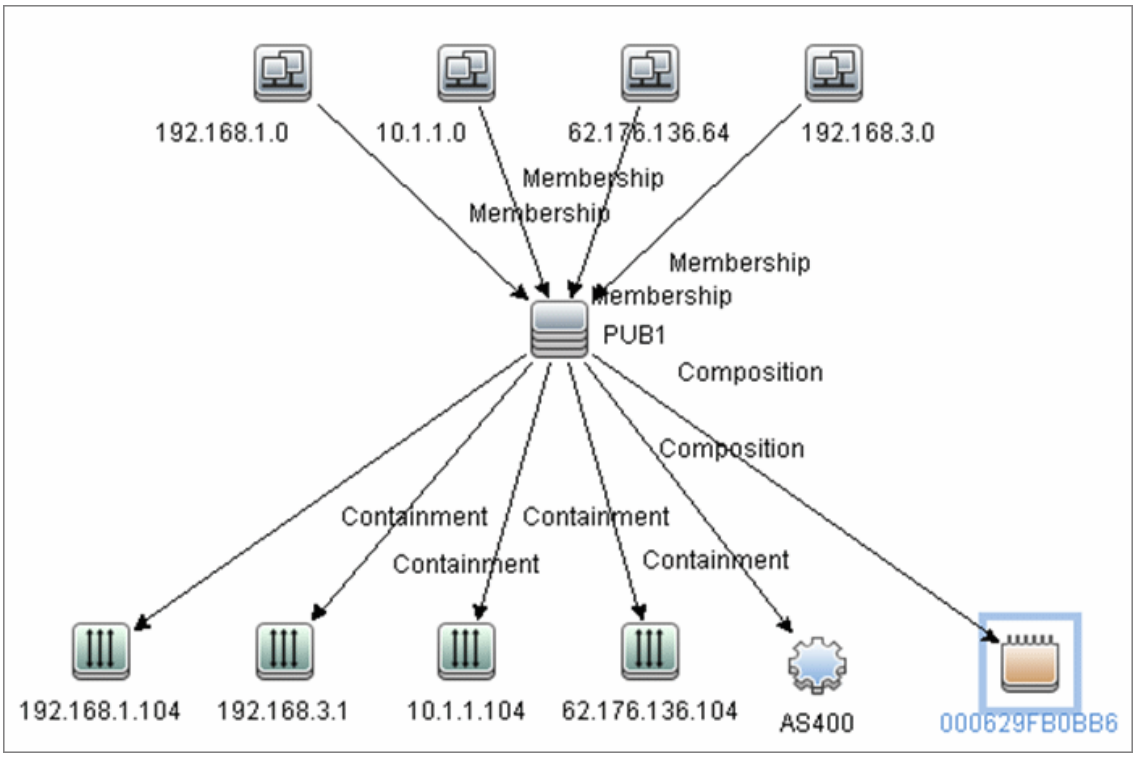
This discovery supports the following versions of AS400:

- V4R2M0
- V3R2M1
- V3R2M0
- V4R5M0
- V5R3
- V6R1

Topology

The following image displays the topology of the AS400 Host discovery with sample output:

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



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" on page 82.

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**

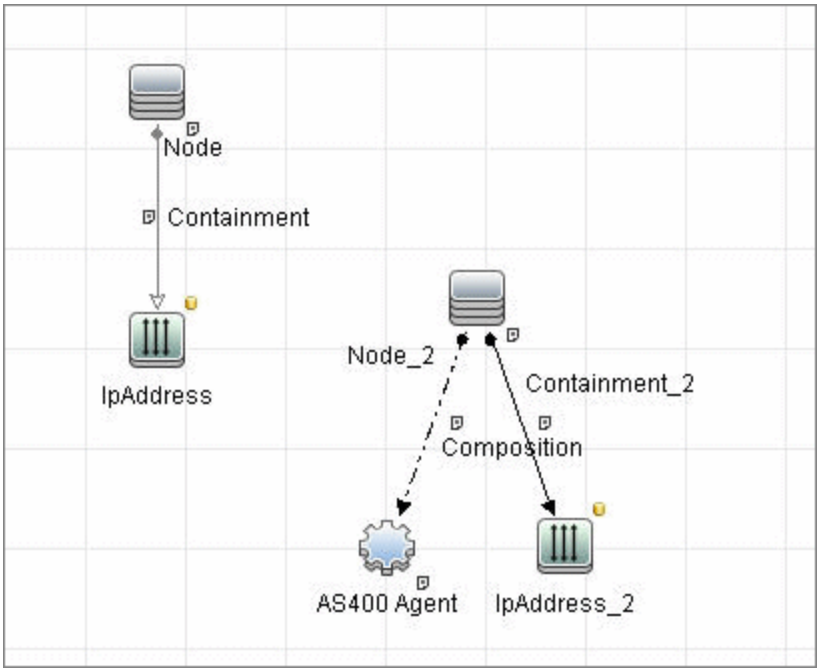
   Activate the **Host Connection to AS400** discovery job.

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

Host Connection to AS400 Job

Trigger Query

Trigger CI:ip_address



Discovered CITs

The following CITs are discovered:

- **IpAddress**
- **AS400Agent**

- **Interface**
- **IpSubnet**
- **Composition**
- **Containment**
- **Membership**
- **Node**
- **Parent**

Note: To view the topology, see "Topology" on page 846.

Chapter 66

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 "Parameters" on page 853.

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" below
- 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 852

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 "Parameters" on page 853.

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" on page 82.

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 Resources and Applications by Shell** job.

   d. Run the **DNS Zone by Nslookup** job.

      For details on running jobs, refer to "Discovery Control Panel" 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 "Parameters" on next page.

2. **Run the discovery**

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

   b. Run the **TCP ports** job.

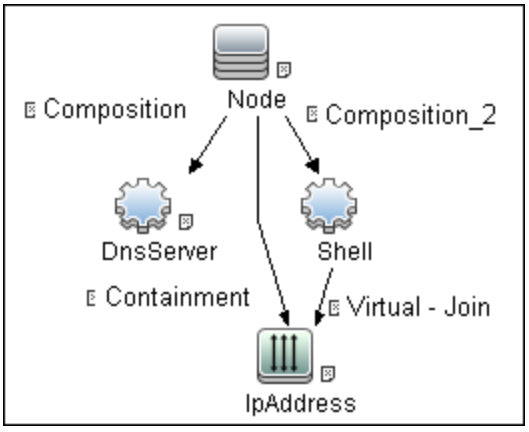
   c. Run the **DNS Zone by DNS** job.

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

DNS Zone by Nslookup Job

This section includes the following:

Trigger Query



- **CI Attribute Conditions**

CI	Attribute Value
Shell attributes	`NOT Reference to the credentials dictionary entry is null`
IP attributes	`NOT IP Probe Name is null`

Adapter

- Input Query

- **Triggered CI Data**

Name	Value
credentialsId	Shell credentials
ip_address	Shell IP address

- **Parameters**

  The adapter includes the following parameters:

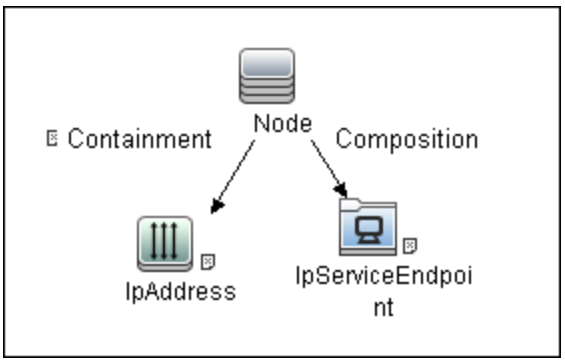
Parameter	Description
isOutOfRangeIpReported	**False**: The IP is not reported if the IP address is out of the Probe's range. **True**: The IP is reported even if the IP address is out of the Probe's range. The default value is **false**.
reportBrokenAliases	**True**: aliases that do not include a canonical 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**.
zoneList	A comma-separated list of zones is an optional attribute for the **DNS Zone by Nslookup** job and mandatory for the **DNS Zone by DNS** job. (If it is not set, an error is raised.) The list provides the names of zones that should be transferred. The default value is an empty value.

- **Created/Changed Entities**
  - The DNS_Zone adapter parameters.
  - The DNS Zone by Nslookup job
  - The DNS Record class (new)

DNS Zone by DNS Job

This section includes the following:

Trigger Query

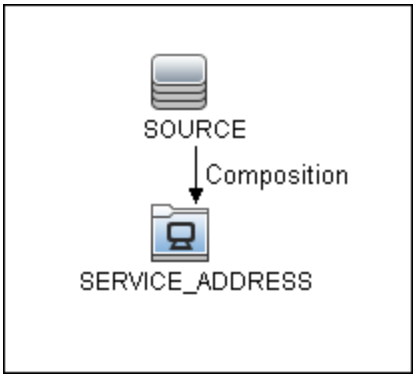


- **CI Attribute Condition**

CI	Attribute Value
IpServiceEndpoint attribute	`Name Equal dns AND NOT IP address is null`

Adapter

- **Input Query**



- **Triggered CI Data**

Name	Value
ip_address	Shell IP address

- **Created/Changed Entities**
  - The DNS_Zone_By_Shell adapter parameters
  - The DNS Zone by Shell job

- The Network – DNS module
- The dns_service Trigger query
- The DNS Record class (new)

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 67

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 stores, 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" on page 82.

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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Host Connection by PowerShell Job

This section includes:

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
```

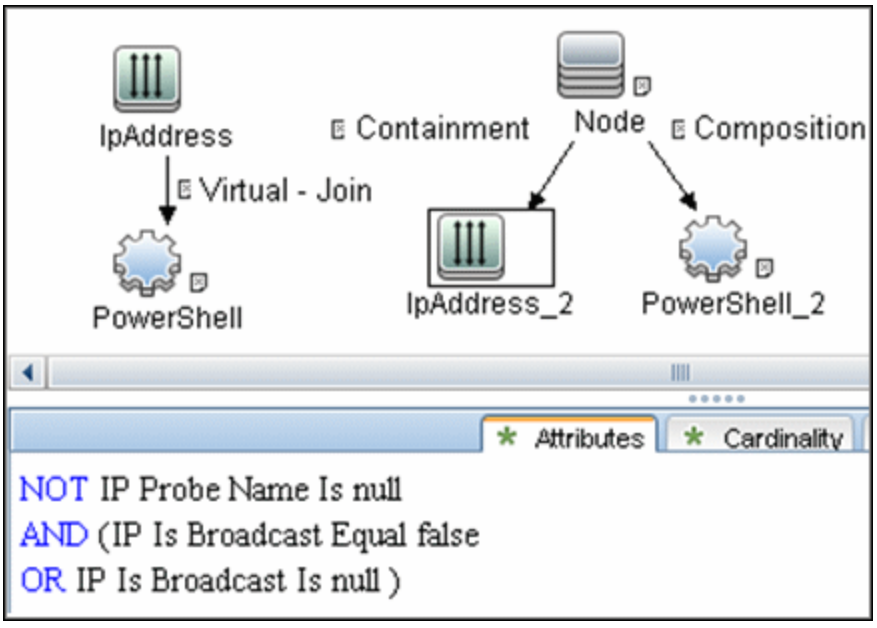
```
IPAddress :
IPSubnet :
MACAddress : 00:16:76:BE:7E:DD
Caption : [00000014] Teefer2 Miniport
Description : Teefer2 Miniport
DhcpEnabled : False
IPAddress :
IPSubnet :
MACAddress : 4A:6F:20:52:41:53
```

- **Mapping**

  The output of this command is used to fill in the attributes of the CIs:

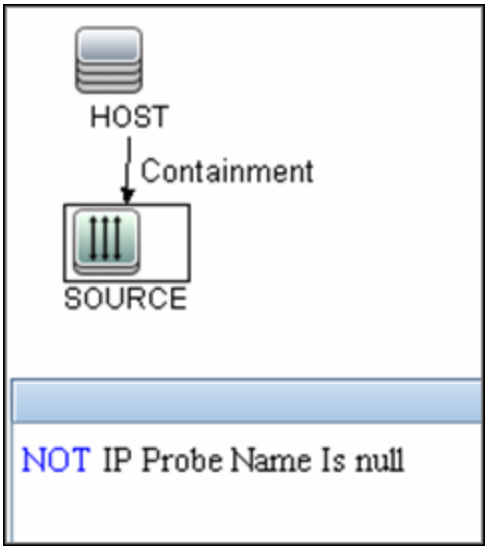
Command Output Attribute	CI Type	CI Attribute
Description	Network Interface	Interface description
DhcpEnabled	Network Interface	DHCP Enabled
IPAddress	IP	IP address
IPSubnet	IP	IP Network Address
MACAddress	Network Interface	Interface MAC Address

Trigger Query



Adapter

- **Input query:**



- **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	
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 please see "HOW TO ENABLE REMOTING FOR ADMINISTRATORS IN OTHER DOMAINS" at http://technet.microsoft.com/en-us/library/dd347642.aspx.

Chapter 68

Host Resources and Applications Discovery

This chapter includes:

Overview

The **Host Resources and Applications** 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 and Applications by Shell/SNMP/WMI** job:

- 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 and Applications by Shell** job also gather connectivity information (either by running `netstat` commands or the **lsof** command).

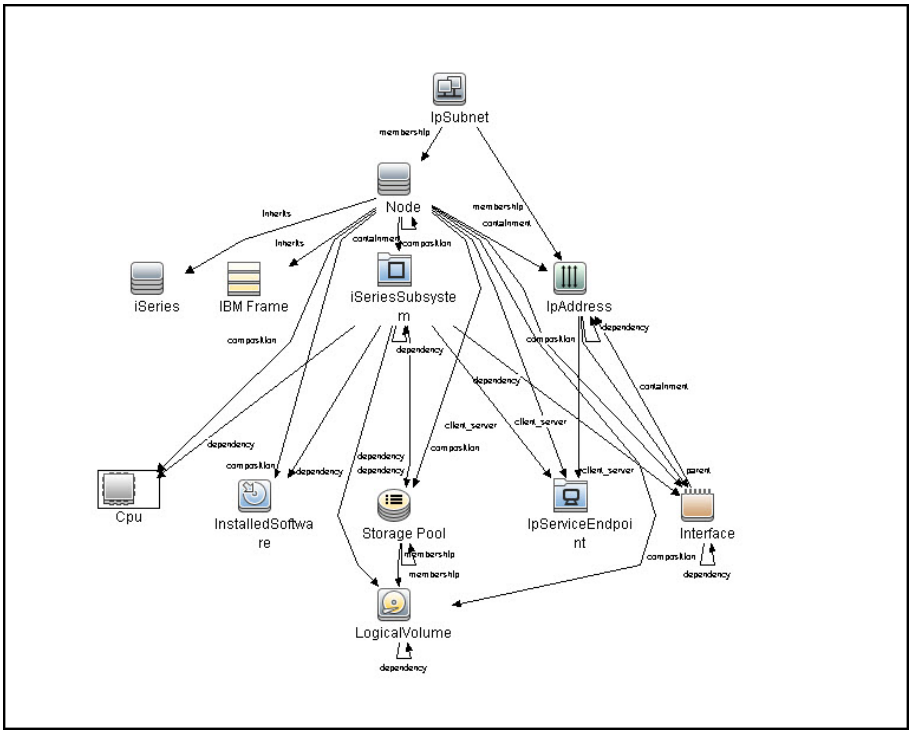
The relationships between processes and the relevant IP Service Endpoint (server port) can be discovered on Windows 2003 and Windows XP, SunOS, Hewlett-Packard UniX (HP-UX), AIX, and Linux operating systems.

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 880.



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:

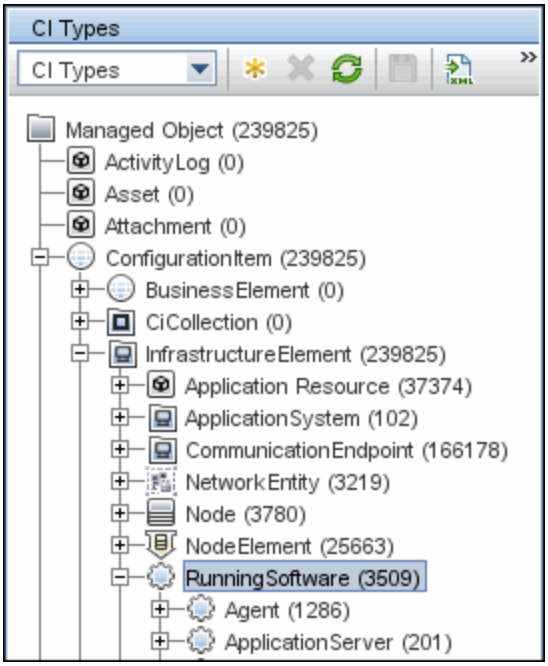
   - NTCMD protocol
   - SNMP protocol
   - SSH protocol
   - Telnet protocol
   - WMI protocol

   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" on page 82.

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:



3. ## Run the Host Resources and Applications by Shell/SNMP/WMI discovery

   In the Discovery Control Panel window, activate the relevant **Host Resources and Applications by Shell/SNMP/WMI** job.

   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. The jobs are scheduled to run every day.

4. ## Run the Software Element CF by Shell discovery

   In the Discovery Control Panel window, activate the **Software Element CF by Shell** job. This job retrieves the running software's configuration file and maps the file to the correct application by referring to the **`applicationsSignature.xml`** file. The triggered CIs are running software that have Shell running on their host and that include a configuration file definition that matches the definition in the **`applicationsSignature.xml`** file.

   For an example on discovering Oracle configuration files, see "Discover Running Software – Scenario" in the *HP Universal CMDB Data Flow Management Guide*.

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

How to Revert to Previous Method of Discovering Installed Software

The Host Resources and 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 **DiscoveryProbe.properties** in the **defaultMaxJobThreads** parameter.

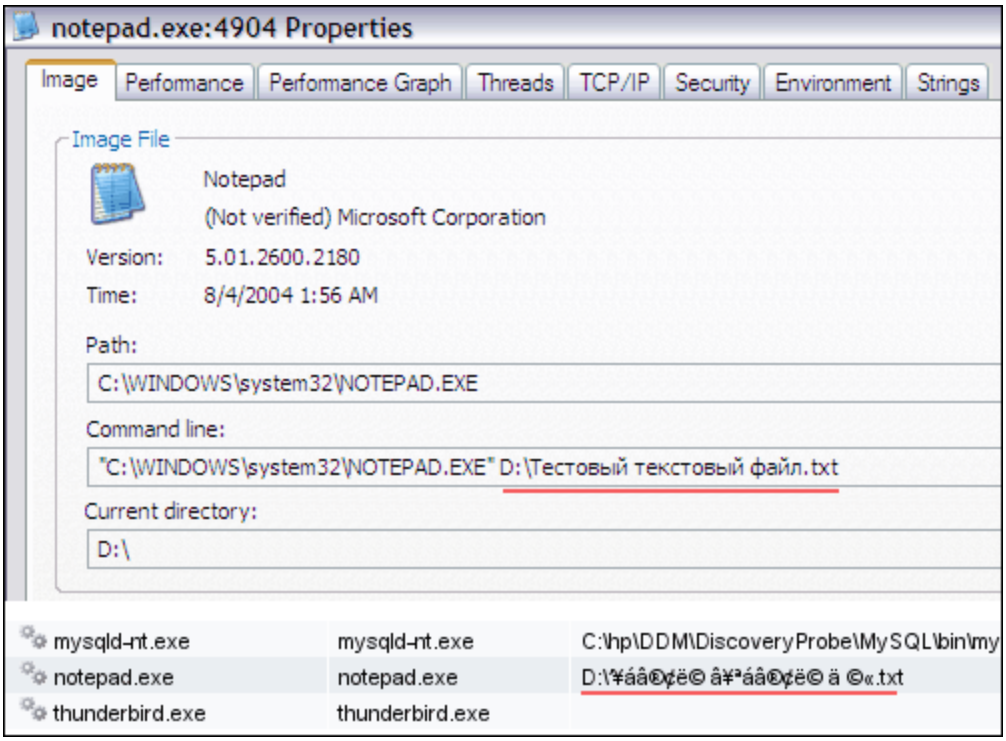
- **regularPoolThreads.** The maximum number of worker threads allocated to the multi-threaded activity (the default is 50).
- **priorityPoolThreads.** The maximum number of priority worker threads (the default is 20).

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:



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 and Applications by Shell job

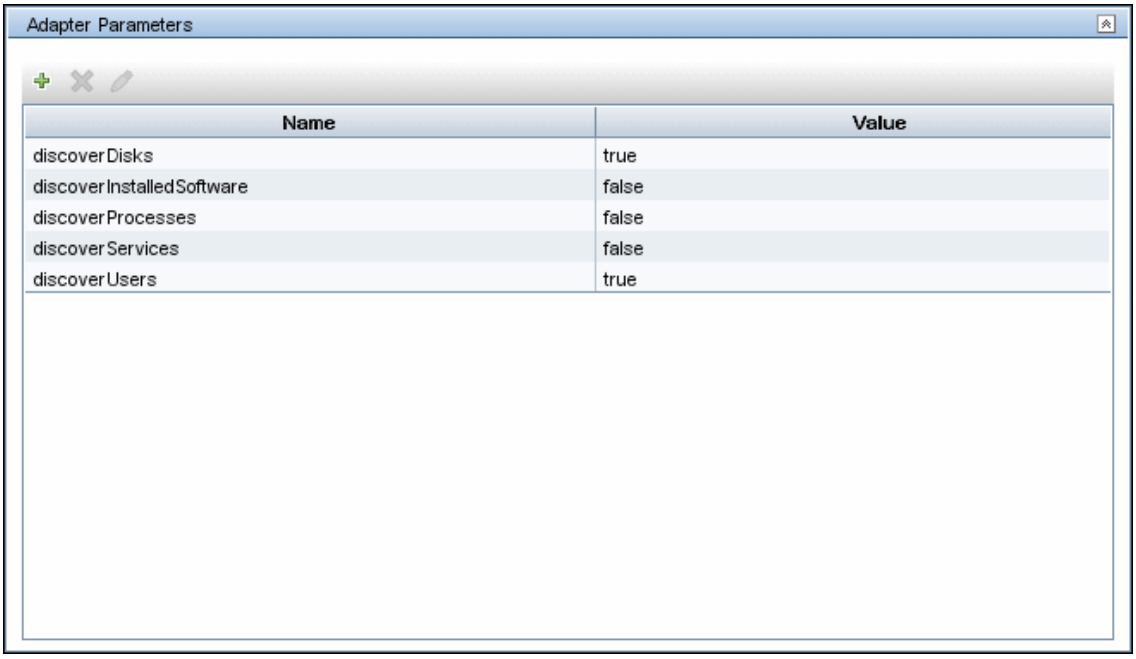
For details, see "Adapter Parameters Pane" in the *HP Universal CMDB Data Flow Management*

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 *.
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. Previously, **false** signified that no processes are discovered.
discoverServices.	• **false**: services are not reported. • **true**. All services are discovered.
discoverShares	**true**: Shared resources are discovered, and **FileSystemExport** CITs are created.
filterP2PProcessesByName (formerly `filterProcessesByName`)	The names of the processes that are not reported. **Default:** system,svchost.exe
lsass.exe,System Idle Process,xCmd.exe	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

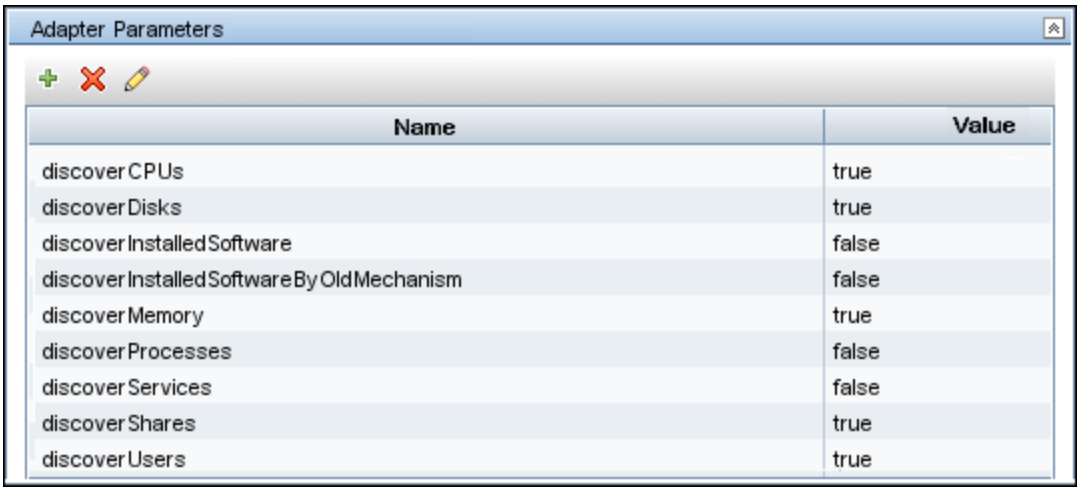
Adapter Parameters for the Host Resources and Applications by SNMP job

For definitions of the parameters, see "Adapter Parameters for the Host Resources and Applications by Shell job" on previous page.



Adapter Parameters for the Host Resources and Applications by WMI job

For definitions of the parameters, see "Adapter Parameters for the Host Resources and Applications by Shell job" on page 877.



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

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*.

Note: To view the topology, see "Topology" on page 874.

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 69

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" on page 82.

   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 and Applications by PowerShell** job to discover the host resources topology.

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

Host Resources and Applications by PowerShell Job

This section includes:

Commands

This section describes each of the commands used by Host Resource and Application by PowerShell discovery.

- **Shared Resources Command**

```
wmic path Win32_Share where "Path <> ''" get Description, Name, Path
```

- **CPU Commands**
  - For Windows 2008 only:

```
wmic path Win32_Processor get DeviceId, MaxClockSpeed,
Manufacturer, LoadPercentage, Name, NumberOfCores
```

  - For Windows 2008 only:

```
wmic Win32_Processor get DeviceId, MaxClockSpeed, Manufacturer,
LoadPercentage, Name, SocketDesignation
```

- **File System Command**

```
wmic logicaldisk get ProviderName, deviceId, driveType, freespace,
size
```

- **Memory Commands**
  - Physical Memory

```
wmic path Win32_PhysicalMemory get Capacity
```

  - Swap Memory

```
wmic PAGEFILESET GET MaximumSize
```

- **Process Command**

```
wmic process get commandLine, creationdate, executablepath, name,
processId
```

- **User Command**

```
wmic path Win32_UserAccount WHERE Domain = '<domainName>' get
Description, Disabled, Domain, FullName, Lockout, Name, SID
```

- **Installed Software Commands**
  - Windows registry query - 64-bit machine key:

```
HKEY_LOCAL_
MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion
```

  - Windows registry query - 32-bit machine key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
```

  - wmic command:

```
wmic path Win32_Product get identifyingNumber, installDate,
installLocation, name, vendor, version
```

- **Service Command**
  - wmic command:

    ```
    wmic service get AcceptPause, Description, DisplayName, Name,
    PathName, ServiceType, StartMode, State
    ```

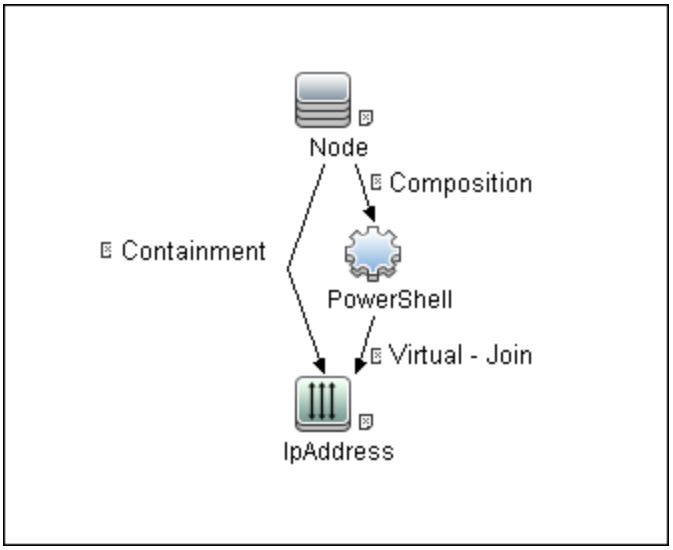
  - Windows registry query:

    ```
    reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services /S|
    findstr "%s" | findstr /V "HKEY_LOCAL_
    MACHINE\\SYSTEM\\CurrentControlSet\\Services\\.*\\.*" | findstr /V
    "Types"
    ```

- **TCP Command**

  ```
  netstat -noa
  ```

Trigger Query

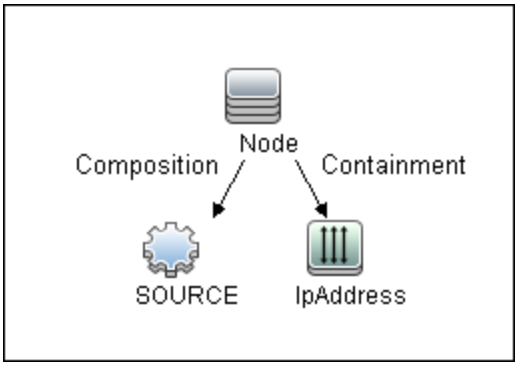


Adapter

- **Input CIT**

  PowerShell

- **Input TQL Query**



Used Scripts

Hostresource_dis_powershell.py

Note: This job may also use library scripts supplied with the AutoDiscoveryContent package.

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**

Created/Changed Entities

Entity Name	Entity Type	Entity Description
Host Resources By PowerShell	Job	New topology job
Host Applications By PowerShell	Job	New topology job
Host Resources and Application Dependency	Module	Discovery module
PowerShell_HR_All	Adapter	Discovery adapter
Host_powershell	TQL	Trigger Query
Hostresource_dis_powershell	Script	Discovery entry point

Chapter 70

iSeries by EView Discovery

This chapter includes:

Overview

The iSeries 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 iSeries by EView Discovery jobs.

Areas of Discovery

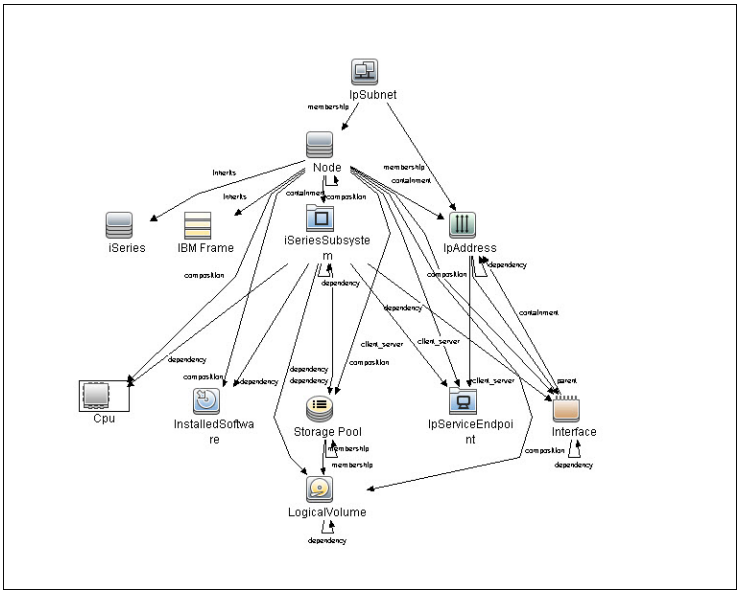
- **iSeries Resources**
  - Local Storage with ASPs
  - Memory
  - Lpars
  - CPUs
  - Network Connectivity
  - Installed Software
  - Selected System Values
  - Subsystems
  - Active Jobs
- **iSeries Objects**
  - Job Queues
  - Output Queues
  - Libraries
  - Program Objects

Supported Versions

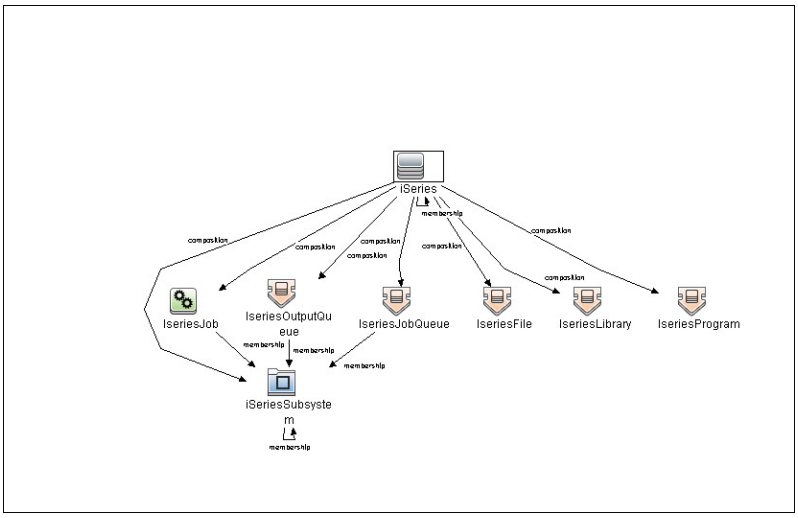
UCMDB Version	iSeries Version
9.x	OS/400 releases V5R1M0 and above

Topology

iSeries Resources



iSeries 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 **iSeries Connection** job to discover the target iSeries host .

   c. Run the **iSeries 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 **iSeries Objects** job to discover object information from the iSeries lpar, such as queues, jobs, program objects, and libraries.

iSeries Connection Job

Input CIT

• Probe

Used Scripts

- eview400_connection.py
- eview400_lib.py

Discovered CITs

- as400_node
- composition
- containment
- eview
- ip_address

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

iSeries Resources 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

- as400_node
- client_server
- composition
- containment
- cpu
- dependency
- ibm_pseries_frame
- installed_software
- interface
- ip_address
- ip_service_endpoint
- ip_subnet
- 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
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

iSeries Objects 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

- as400_node
- composition
- iseries_job
- iseries_jobqueue
- iseries_library
- iseries_outqueue
- 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 Ipar. **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

Chapter 71

Layer 2 Discovery

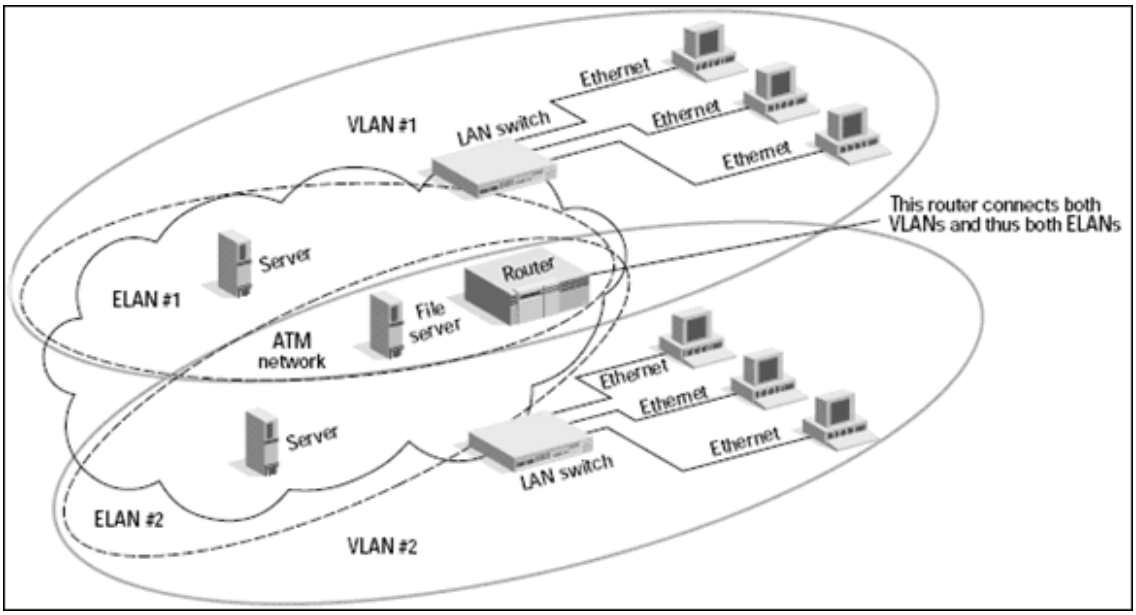
This chapter includes:

Overview

The Layer 2 package discovers the Layer 2 topology that includes the switches tree topology (the backbone links between the switches) and also the end user connections to the switch-ports (the Layer 2 CIs between a switch and a host).

The Layer 2 package is based on the SNMP protocol.

The following image illustrates a router connecting overlapping VLANs/ELANs:



Supported Versions

This discovery supports any switch that supports the standard bridge (MIB - RFC 1493).

[ES CR 71173 CP 11 removed this]

How to Discover Layer 2 Objects

This task describes how to discover Layer 2 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" on page 82.

2. **Prerequisite - Other**

   - All network connection jobs should finish running before you activate the Layer 2 jobs.

- 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 Layer 2 topology.

3. ## Run the discovery

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

Activate the jobs in the following order:

a. Activate the **Host Networking by SNMP** job. This job discovers host networking topology using SNMP route and system tables. As a result of this run, DFM saves SNMP CIs to the CMDB. You should run this job on all SNMP agents on the switches that were discovered in the environment. The to-be discovered Layer 2 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.)

> **Note:** Layer 2 discovery is based on the connection jobs for the following reasons:
>
> - The Layer 2 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 Layer 2 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 Layer 2 module.

b. 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.

c. 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**. For details on the SNMP protocol parameters, see SNMP Protocol in the *HP Universal CMDB Data Flow Management Guide*.

The `SNMP_Net_Dis_VMS_catalyst.py` script retrieves the Base MAC table and Port number If Index table.

d. 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 the way they retrieve the community string:

- **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 previous page, based on the triggered VLAN number.

> **Note:**
>
> - When the VLANs by SNMP job runs, it discovers Layer 2 topology that is relevant to the discovered VLAN only.
> - Bridge Layer 2 discovery. If a machine has no VLANs, discovery is triggered on the bridge of the switch. DFM retrieves the Layer 2 topology of all the switches.
> - If you dispatch the Bridge Layer 2 job on the bridge of a switch that holds VLANs only, the default VLAN Layer 2 topology is discovered.

e. 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." above.

VLANS by SNMP Job

Discovered CITs

- **Bcast Domain**

- **Composition**
- **ELAN**
- **ELAN-VLAN Map**
- **Membership**
- **PhysicalPort**
- **Vlan**

VLAN ports by SNMP Job

Discovered CITs

- **Bridge**
- **Composition**
- **Containment**
- **Dependency**
- **Membership**
- **PhysicalPort**
- **Vlan**

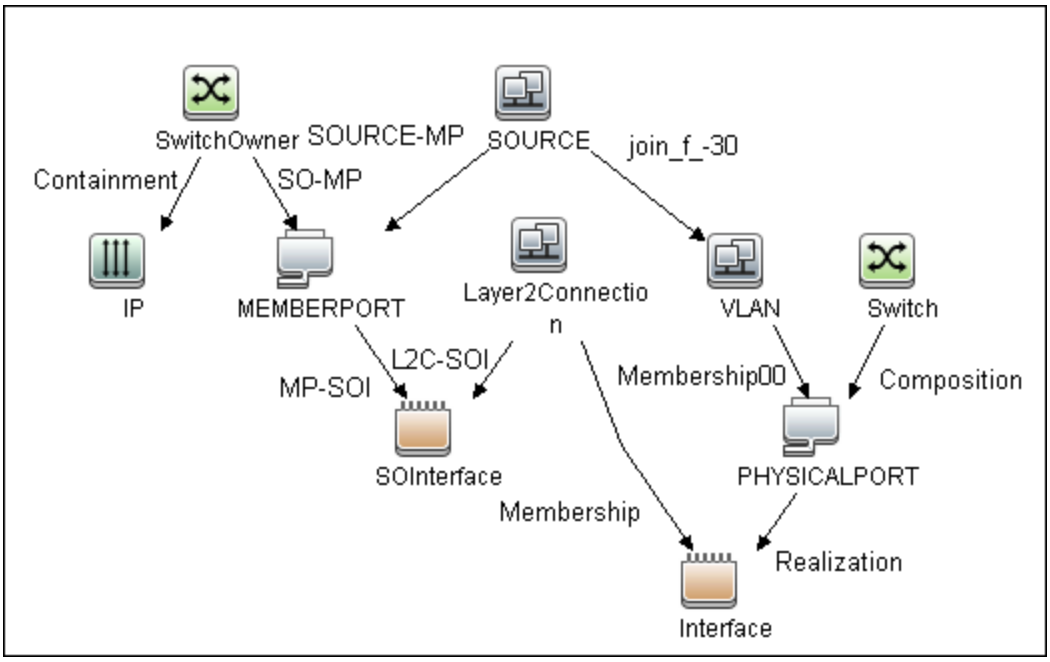
Merge VLANs by VLAN Ports Job

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:

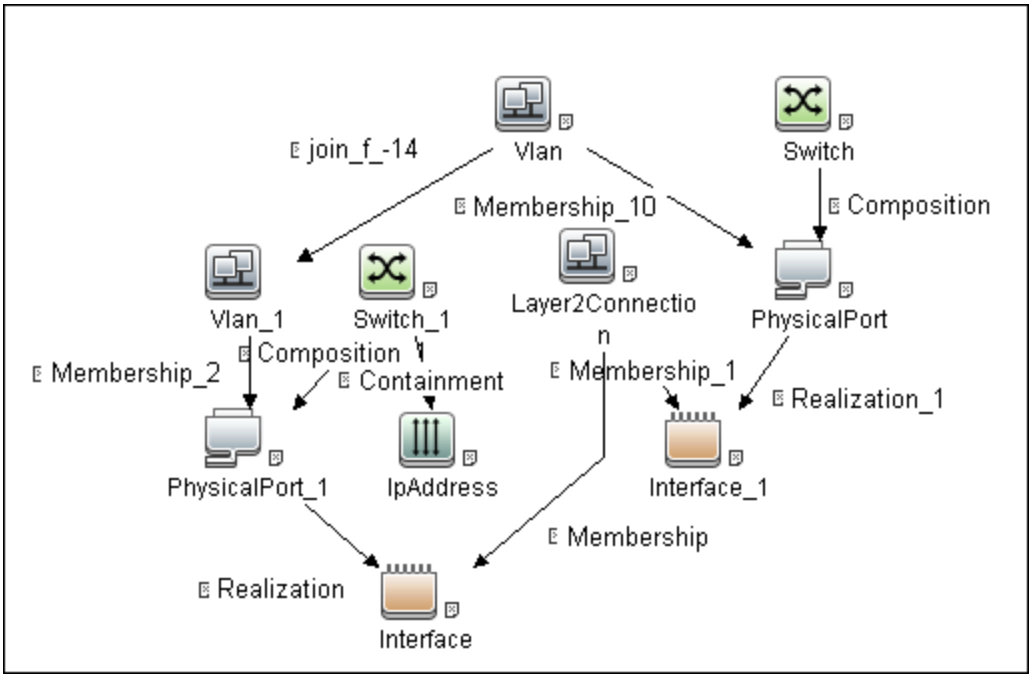
1. The ports which are related to a VLAN are connected by a Layer 2 Connection; and
2. The VLAN id is the same.

- "Input Query" on next page
- "Trigger TQL Query" on next page
- "Triggered CI Data" on page 902
- "Discovered CITs" on page 902

Input Query



Trigger TQL Query



Triggered CI Data

Name	Value
memberId	${MEMBERPORT.root_id}
portId	${PHYSICALPORT.root_id}
vlanId	${SOURCE.vlan_id}

Discovered CITs

- **Membership Link**
- **PhysicalPort**
- **Vlan**

Layer2 Topology Bridge based by SNMP

Discovered CITs

- **Bridge**
- **Composition**
- **Interface**
- **Layer2Connection**
- **Membership**
- **Node**
- **PhysicalPort**
- **Realization**

Layer2 Topology VLAN based by SNMP Job

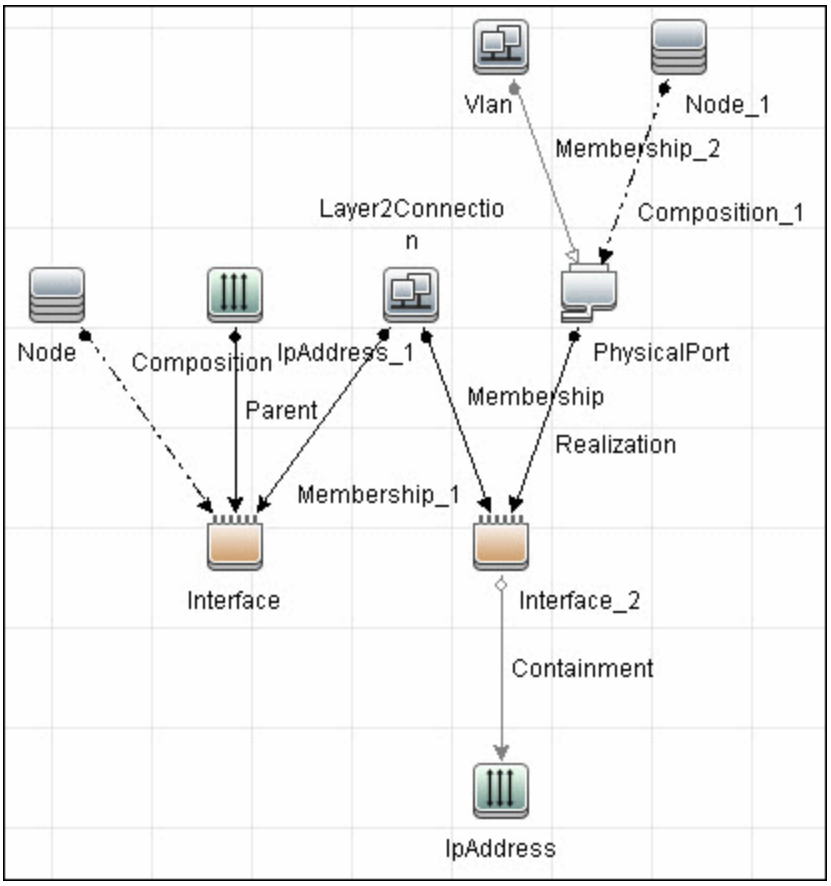
Discovered CITs

- **Bridge**
- **Composition**
- **Interface**
- **Layer2Connection**
- **Membership**

- **Node**
- **PhysicalPort**
- **Realization**

Relationships

- A Layer 2 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 Layer 2 CI, or to a port-switch by a Backbone link.



Troubleshooting and Limitations

This section describes troubleshooting and limitations for Layer 2 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 72

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 **Network - Basic** module uses the following jobs:

- **Host Connection by Shell**. Establishes the connection to remote machines through the SSH, Telnet, and NTCMD protocols. This job discovers host type, OS information, and network connectivity information. For details, see "How to Discover Host Connection by Shell" on next page.
- **Host Connection by SNMP**. Discovers SNMP agents by trying to connect to a machine 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 Host Connection by SNMP" on page 909.
- **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 909.

For details on using a wizard to discover the network, see "Infrastructure Discovery Wizard" in the HP Universal CMDB Data Flow Management Guide.

For information about each job's discovery mechanism, see:

- **Host Connection by Shell.** "Host Connection by Shell Job" on page 910.
- **Host Connection by SNMP**. "Host Connection by SNMP Job" on page 920
- **Host Connection by WMI**. "Discovery Mechanism" on page 924

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

   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" on page 82.

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 "Discovery Control Panel" 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" on page 82..

2. **Run the discovery**

   Run the **Host Connection by SNMP** job.

   For details on running jobs, refer to "Discovery Control Panel" 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" on page 82..

2. **Run the discovery**

   Run the **Host Connection by WMI** job.

   For details on running jobs, refer to "Discovery Control Panel" 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 macines or UNIX-based machines. For details on the DFM processes, see:

- "Windows Processes" on next page
- "UNIX-Based Processes" on page 912

Note:

- DFM tries to connect using the credentials last used for this destination.
- If the credentials do not exist, or if the connection fails, DFM tries to connect by using another protocol in a predefined list of protocols (SSH, Telnet, NTCMD) together with its 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 OS. For details, see:

- "AIX" on next page
- "FreeBSD" on next page
- "HPUX" on page 914
- "LINUX" on page 914
- "OpenBSD" on page 915
- "SunOs" on page 916
- "VMKernel" on page 916

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**).

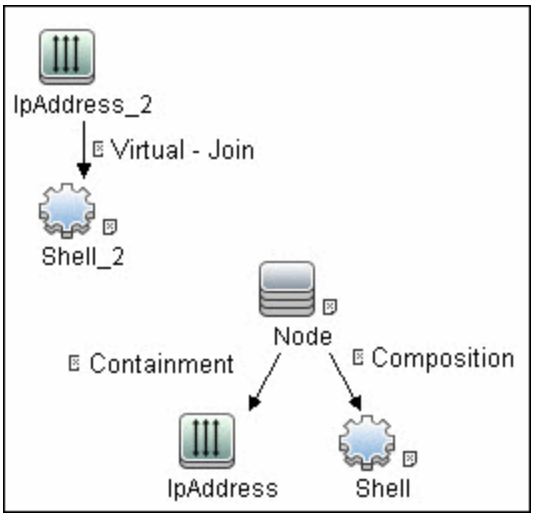
   Full command:`'esxcfg-info | grep \'BIOS UUID\'`
5. The serial number (**esxcfg-info**).

   Full command:`'esxcfg-info -w | grep \'Serial Number\''`
6. The default gateway (**esxcfg-route**).
7. The OS flavor (**vmware**)

   Full command:`'vmware -v'`

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

- **codepage**. The discovered machine codepage. Default: **NA**.
- **language**. The discovered machine language. Default: **NA**.
- **useAIXhwId**. Used to identify IBM AIX machines through their hardware ID. **true**: when used together with SNMP discovery, duplicate hosts may be created. **false**: no AIX LAPR 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

- **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 **xCmd 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

- **Limitation:** If an interface has a MAC address of 0, the job does not report that interface or the IP address assigned to it.

Host Connection by SNMP Job

This subject includes the following sections:

Discovery Mechanism

1. DFM runs through the credentials defined for the SNMP protocol and tries to connect successfully through one of them.
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:

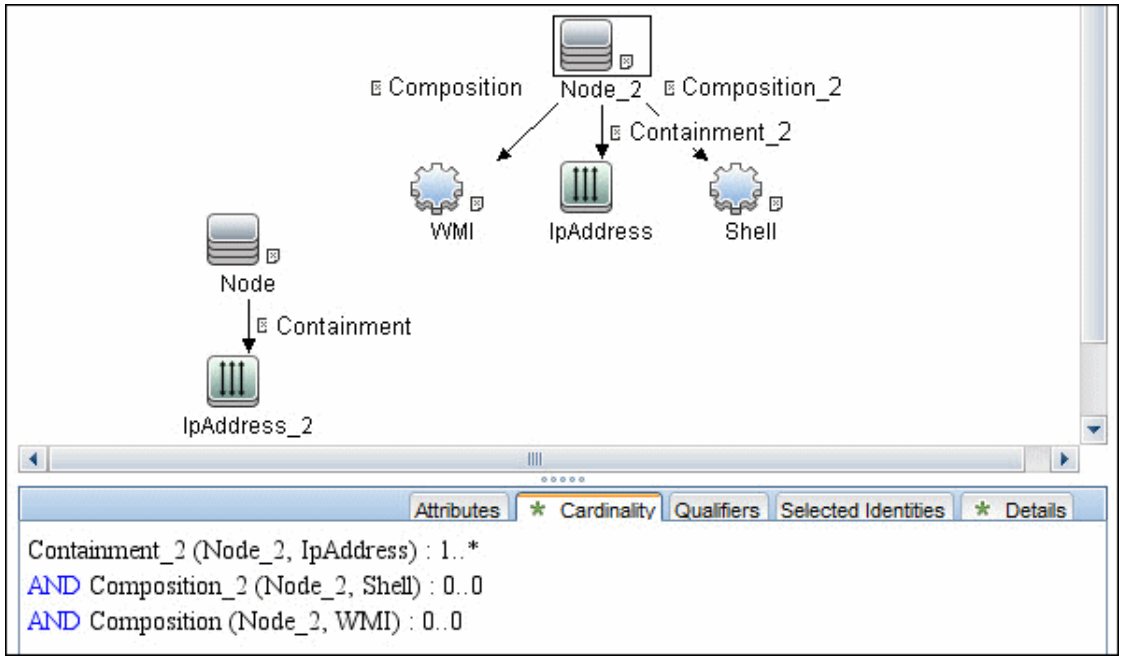
```
OID (1.3.6.1.2.1.2.2.1) - an interface entry containing objects at
the subnetwork layer and below for a particular interface.
```

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
```

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.



- **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

- **ATM Switch**
- **Composition**
- **Containment**
- **Firewall**
- **Interface**
- **IpAddress**
- **IpSubnet**
- **Load Balancer**
- **Mainframe**
- **Membership**
- **Net Device**
- **Net Printer**
- **Noce**
- **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**: The workflow for this job is **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:

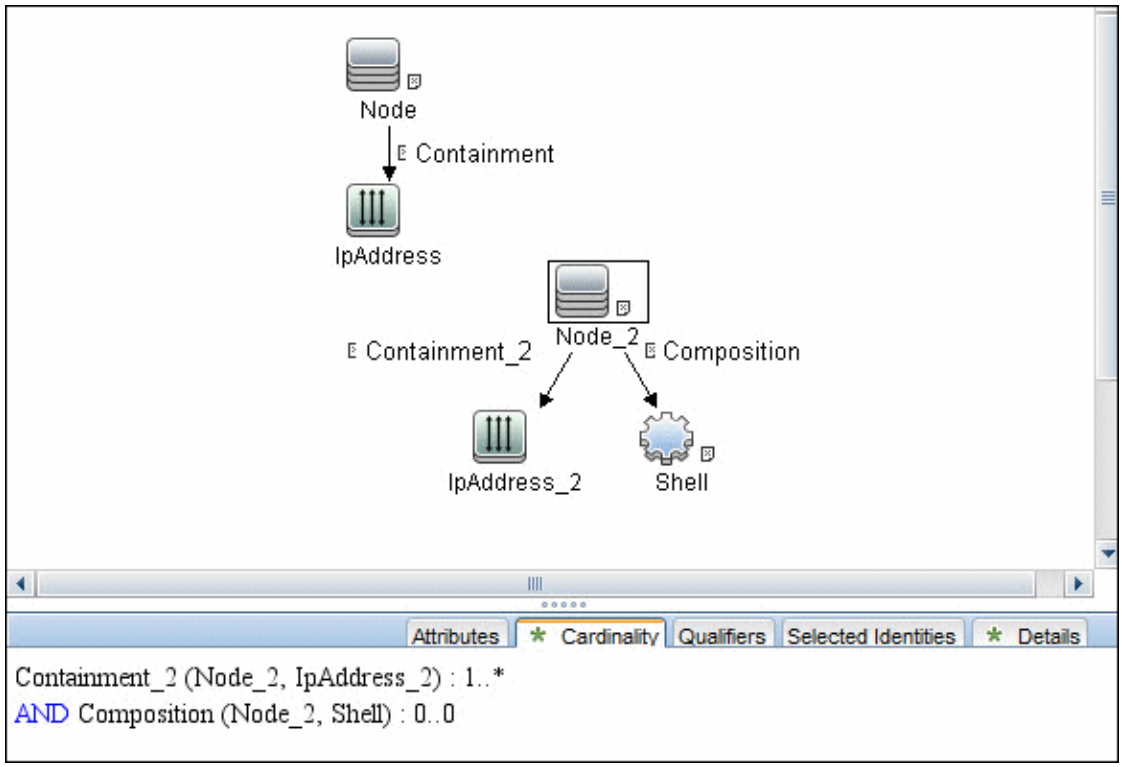
   ```
   SELECT SerialNumber,SMBIOSAssetTag FROM Win32_SystemEnclosure
   ```

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.



- **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**.

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. The Host Fingerprint using nmap job, which relies on the Nmap utility, reports the Nmap accuracy value on the **`host_osaccuracy`** attribute on the Host CI.

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:

- "Prerequisites- Set up protocol credentials" below
- "Prerequisites - Set up Data Flow Probe machine" below
- "Run the discovery" on page 935

1. Prerequisites- Set up protocol credentials

   For credential information, see "Supported Protocols" on page 82.

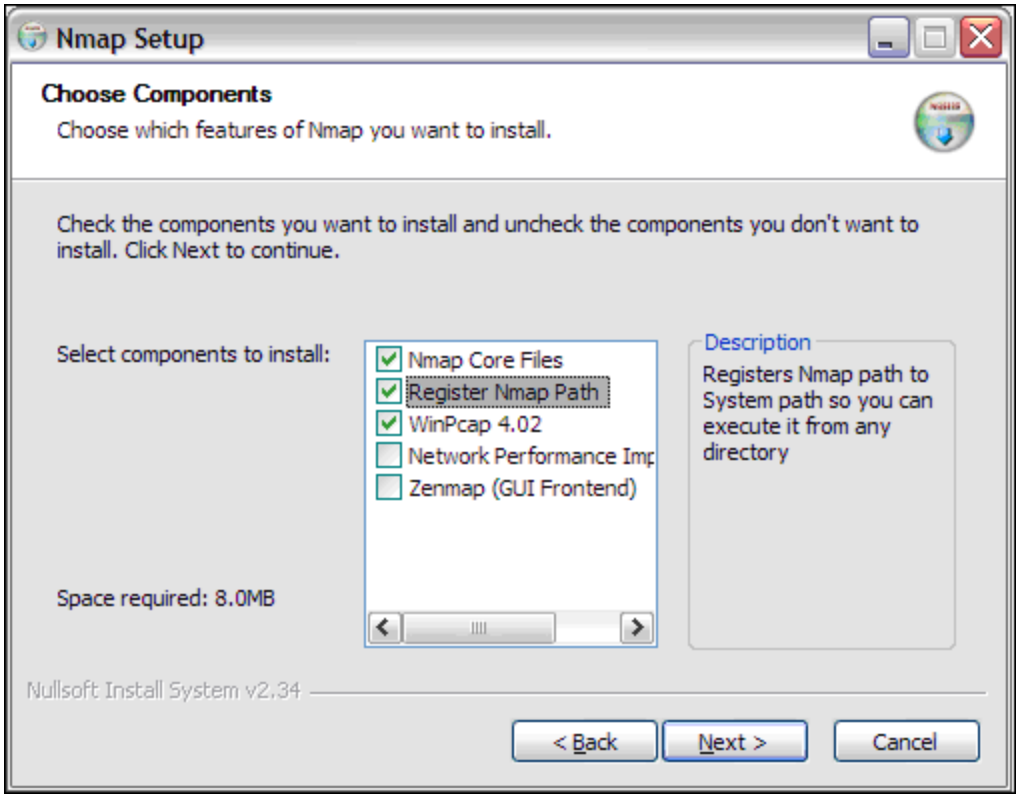
2. **Prerequisites - Set up Data Flow Probe machine**

   Perform the following procedure on every Data Flow Probe machine that is to run the Host Fingerprint using nmap job:

   a. Run **nmap-4.76-setup.exe** from **C:\hp\UCMDB\DataFlowProbe\tools**.

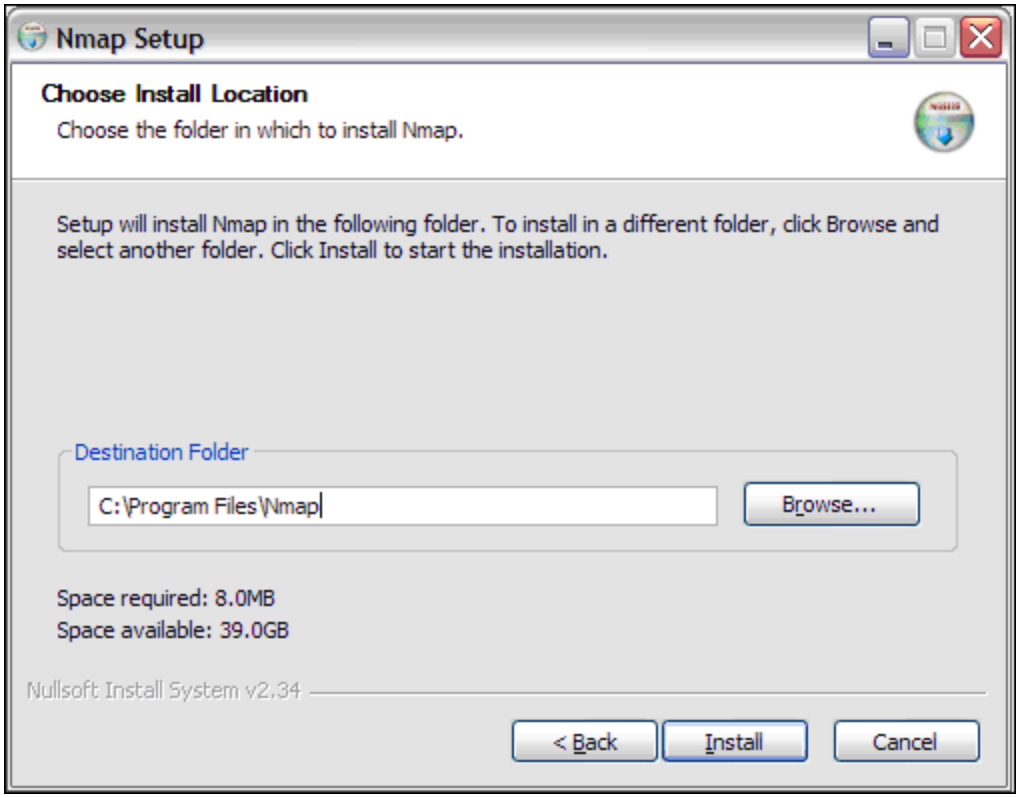
   b. Accept the terms of the license and click **I agree**. The **Choose Components** dialog box opens.

   c. Select **Nmap Core Files**, **Register Nmap Path**, and **WinPcap 4.02**.


Nmap Setup
Choose Components
Choose which features of Nmap you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Nmap Core Files
Register Nmap Path
WinPcap 4.02
Network Performance Imp
Zenmap (GUI Frontend)
Description
Registers Nmap path to System path so you can execute it from any directory
Space required: 8.0MB
Nullsoft Install System v2.34
< Back
Next >
Cancel

d. Click **Next**.

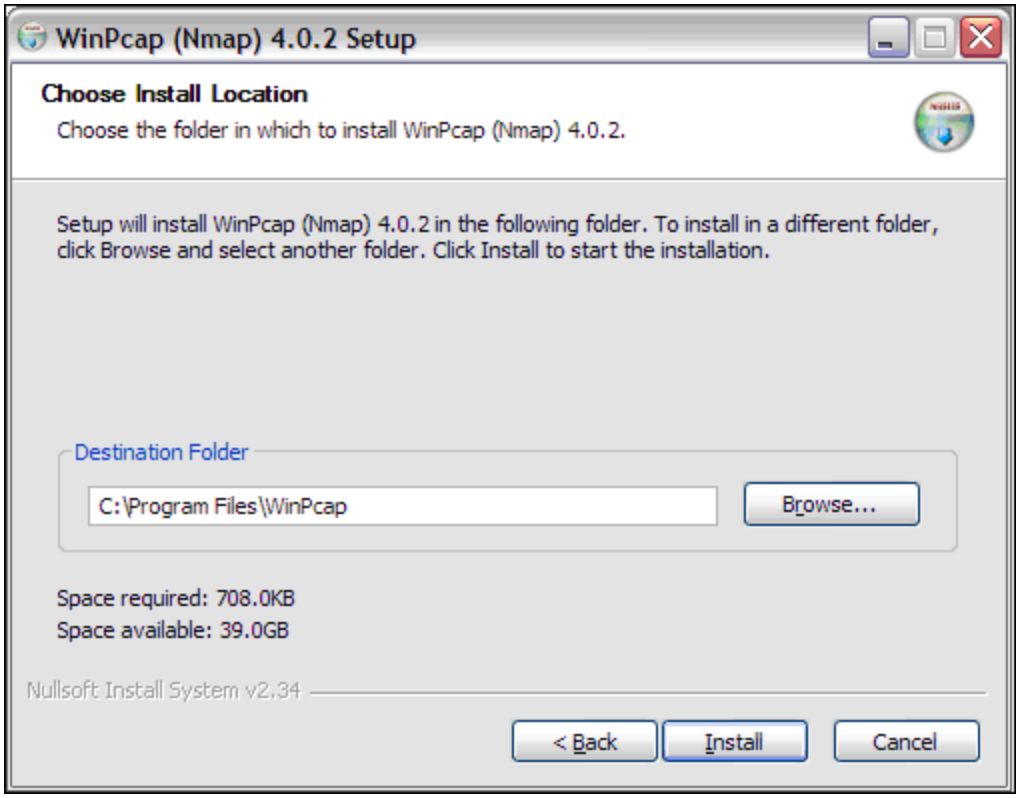
The **Choose Install Location** dialog box opens.



e. Accept the default location or enter another location. Click **Install**.

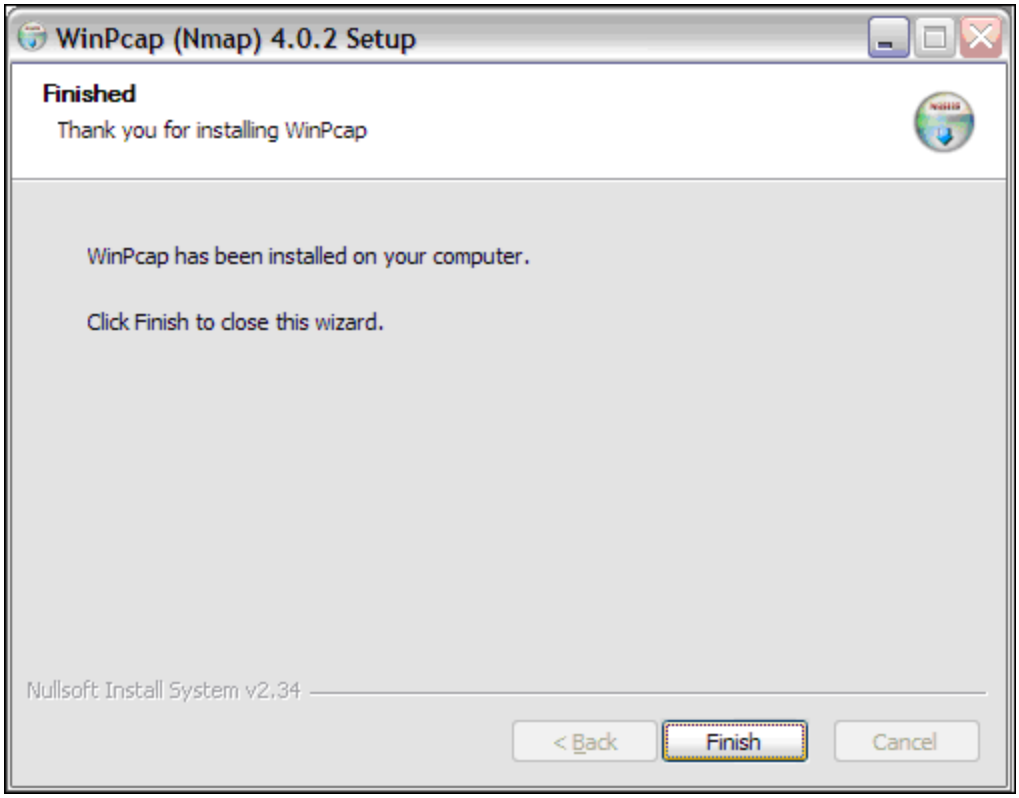
Nmap is installed. The WinPcap installation dialog box opens immediately after the Nmap installation is complete.

f. Accept the terms of the license and click **Next**. The **Choose Install Location** dialog box opens.

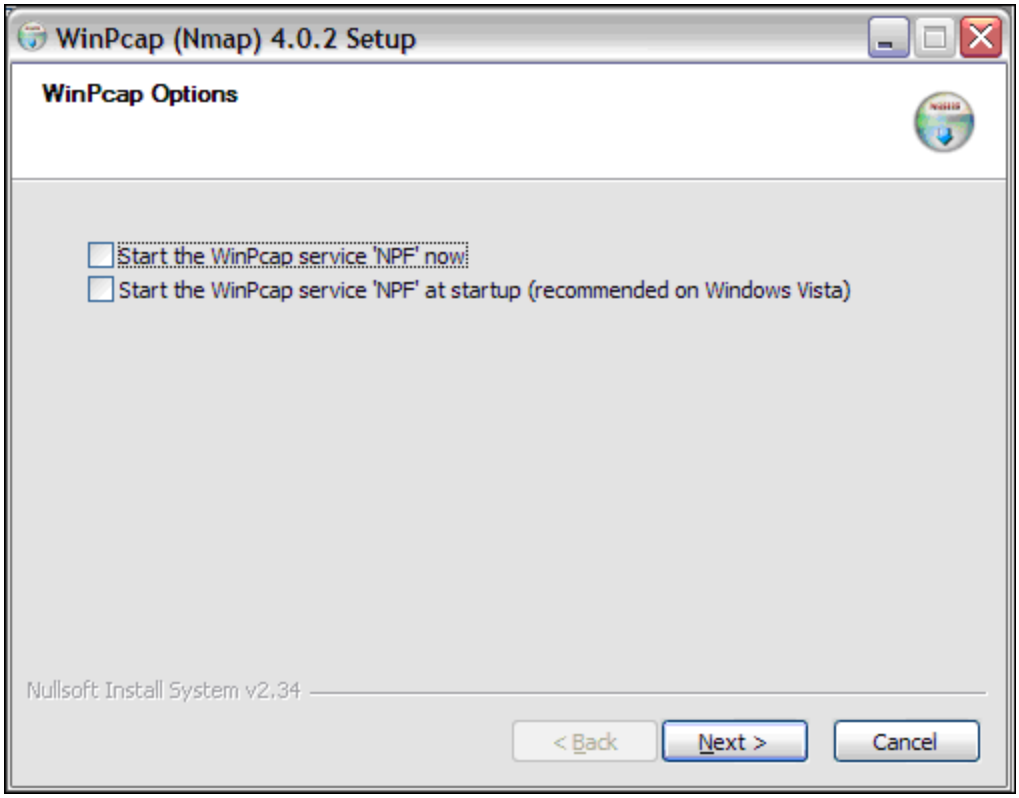


g. Accept the default location or enter another location. Click **Install**.

The Finished dialog box opens.



Click **Finish**. The WinPcap Options dialog box opens.



h. Clear the check boxes and click **Next**.

i. Click **Finish**.

The following software is added to the Data Flow Probe machine:

- Nmap 4.76
- winpcap-nmap 4.02
- Microsoft Visual C++ Redistributable - x86 9.0.21022

To verify, access the **Add/Remove Programs** window.

3. **Run the discovery**

This job is triggered on any discovered IP address.

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

Host Fingerprint Using Nmap Job

Adapter Parameters

To view the adapter parameters: **Discovery Control Panel > Network Discovery >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).
scan_ known_ ports_only	Scans for ports listed in the **portNumberToPortName.xml** file. **Default:** False
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*.

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 930. 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 930.

Part XI: Virtualization

Chapter 74

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

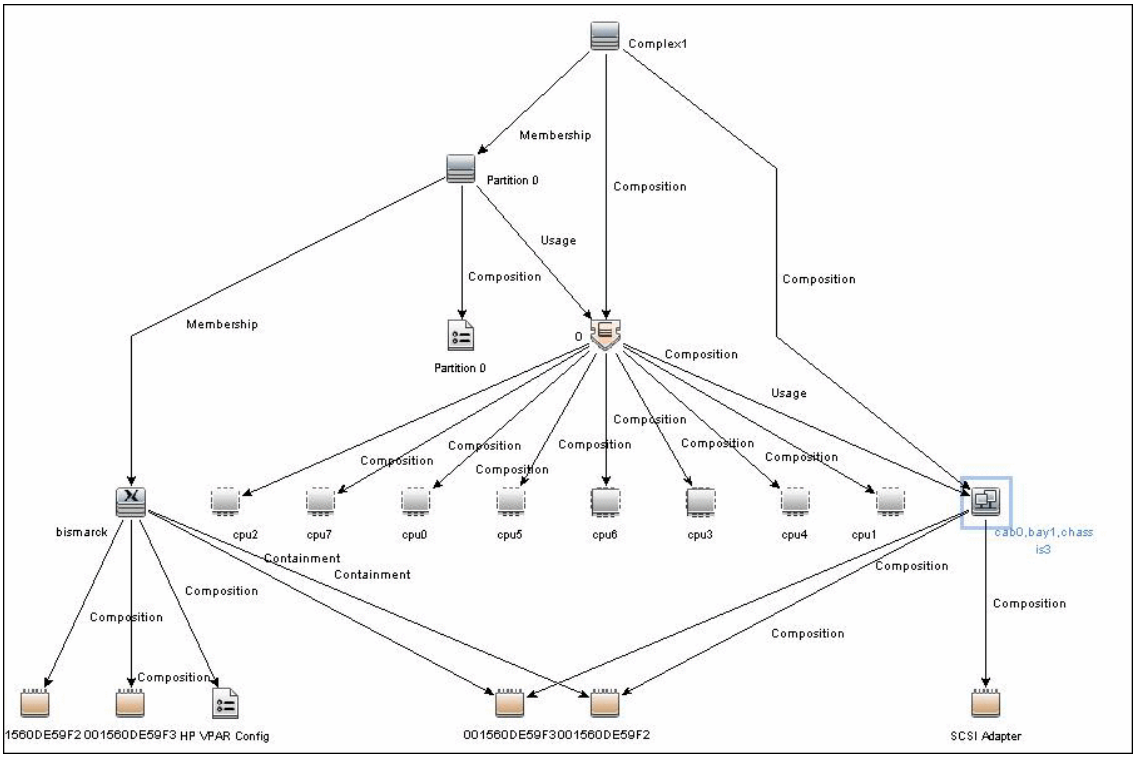
This discovery is relevant for the vPars A.03.xx, A.04.xx, and A.05.xx versions.

This package has been verified on cellular systems with vPars running a HP-UX operating system. Non-cellular systems and vPars running other operating systems are not supported in this version.

Topology

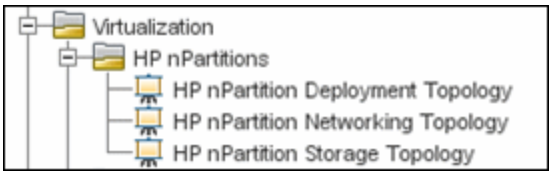
This section includes:

HP vPars and nPars Topology

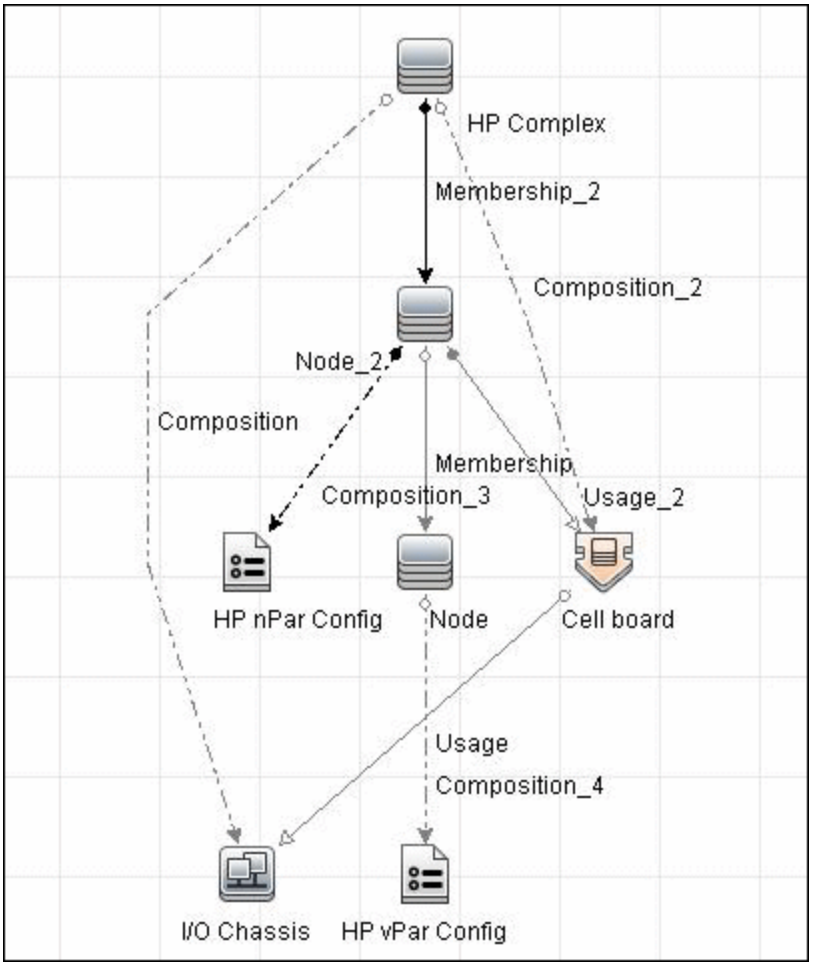


HP nPartitions Topology Views

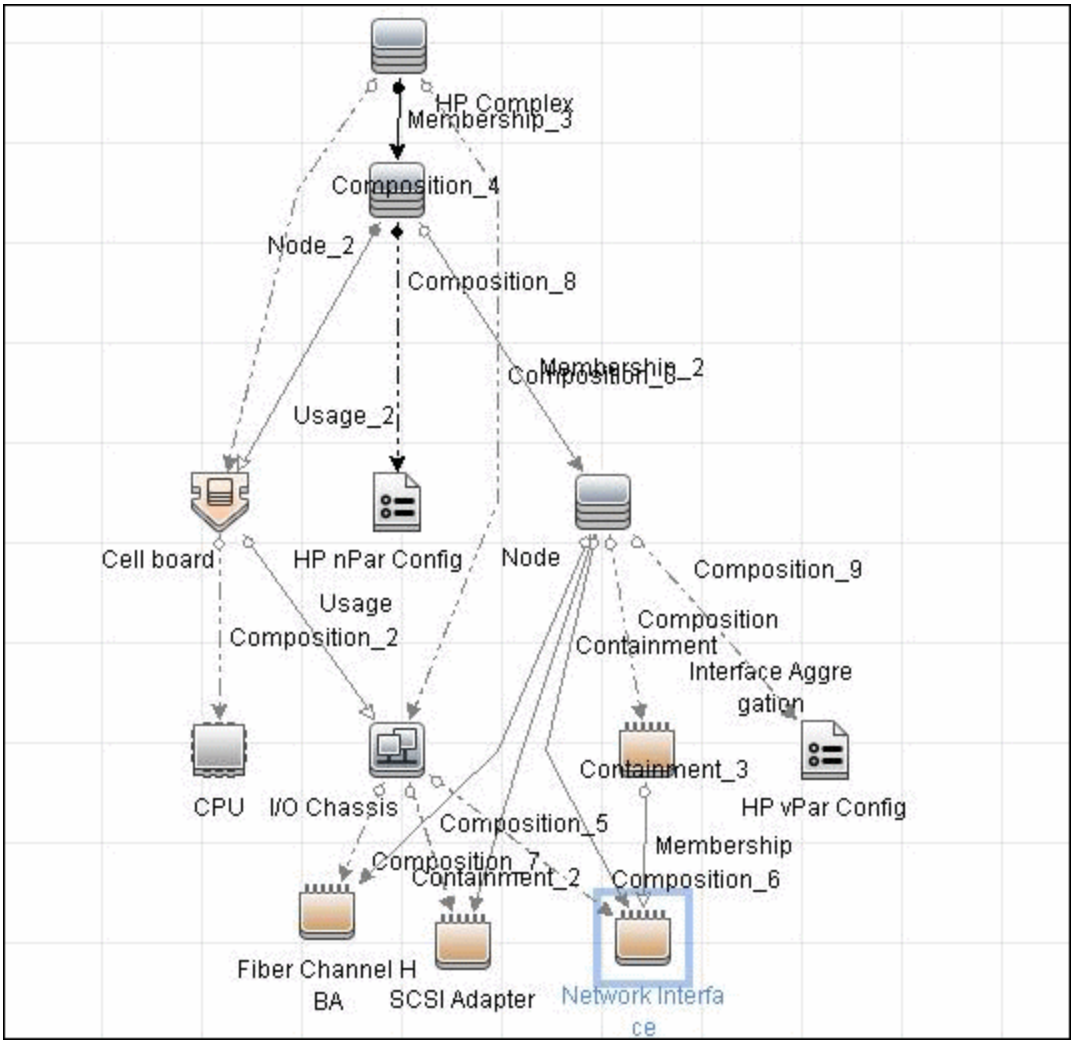
HP nPartitions topology is represented by the following views under the Virtualization module:



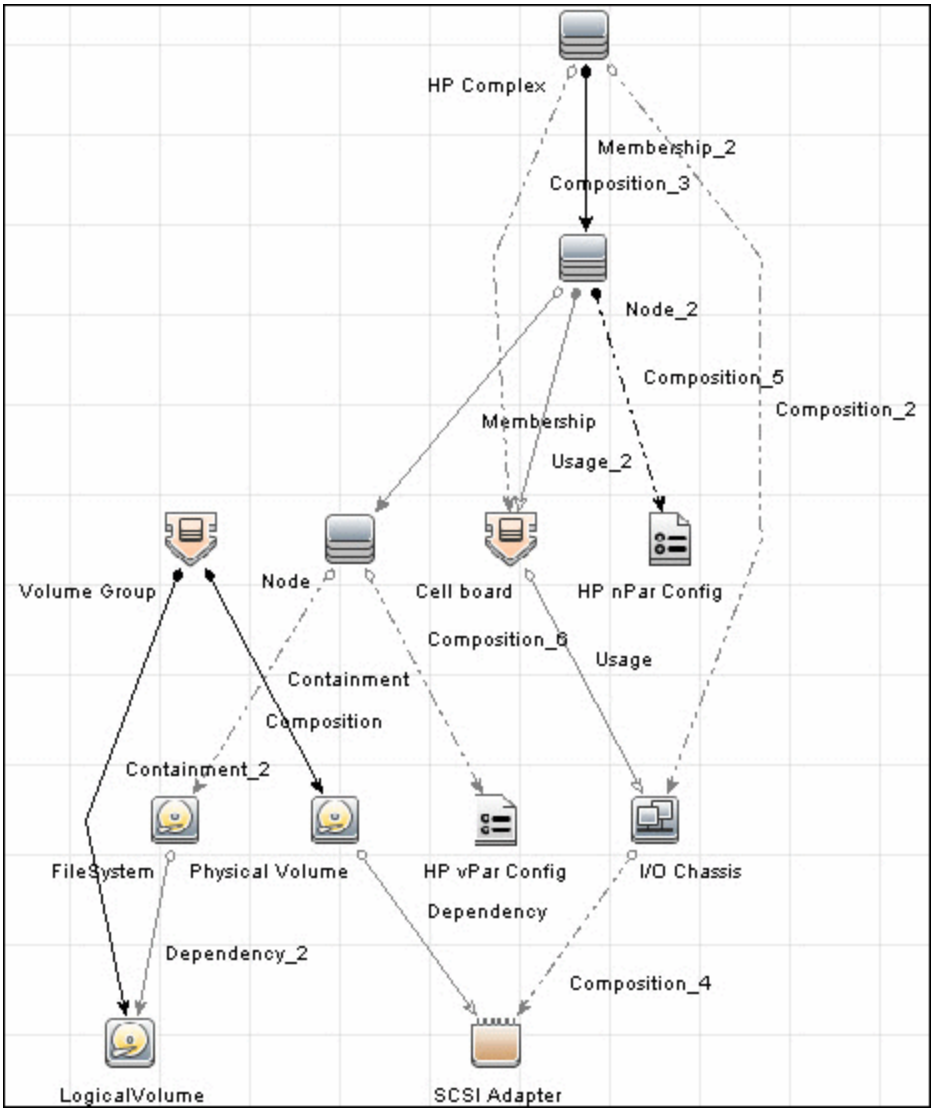
- **HP nPartition Deployment Topology View**
  This view represents the basic virtualization deployment, containing nPars, vPars, cells, and I/O chassis only.


HP Complex
Membership_2
Composition_2
Node_2
Composition
Membership
Composition_3
Usage_2
HP nPar Config
Node
Cell board
Usage
Composition_4
I/O Chassis
HP vPar Config

- **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" on page 82.

2. **Run the discovery**

   For details on jobs, see "Discovery Control Panel – Advanced Mode Workflow" 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 nPartitions by Shell** job.

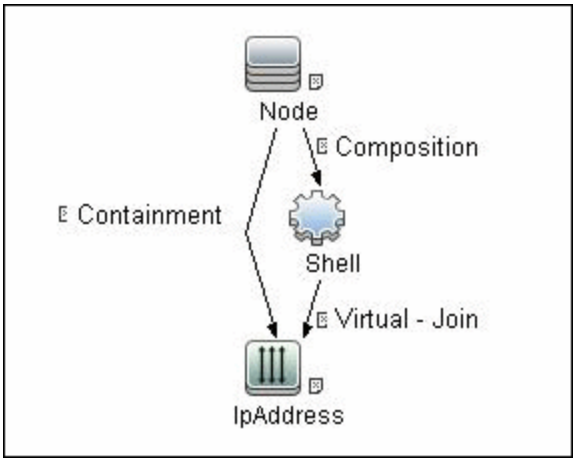
For details on running jobs, refer to "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

HP nPartitions by Shell Job

This section includes:

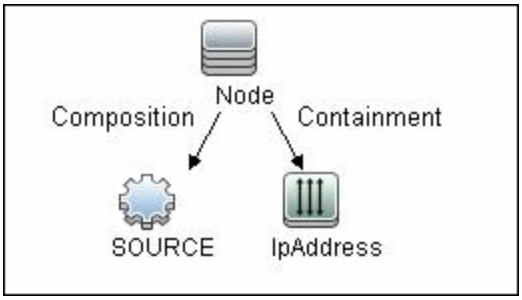
Trigger Query

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



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
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
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 ---

<table>
<tr><td></td><td colspan="2">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</td></tr>
<tr><td rowspan="9">Values taken</td><td colspan="2">Volume group</td></tr>
<tr><td>VG Name > name
VG Write Access > write_access</td><td></td></tr>
<tr><td>VG Status > vg_status
PE Size (Mbytes)</td><td>This value is used to calculate the size of the physical volume</td></tr>
<tr><td colspan="2">Logical Volume</td></tr>
<tr><td>LV Name > name
LV Status > lv_status</td><td></td></tr>
<tr><td colspan="2">Physical Volume</td></tr>
<tr><td>PV Name > name</td><td>Alternate link may also be used. It depends on the output of the ioscan FnkCdisk command.</td></tr>
<tr><td>PV Status > pv_status</td><td></td></tr>
<tr><td>Total PE > pv_size</td><td>This attribute is calculated on the PE Size (Mbytes) value.</td></tr>
</table>

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 -m
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" on page 82.

- 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.

  Failed to discover Fibre Channel adapters.

  Failed to discover Network cards.

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

Chapter 75

Hyper-V Discovery

This chapter includes:

Overview

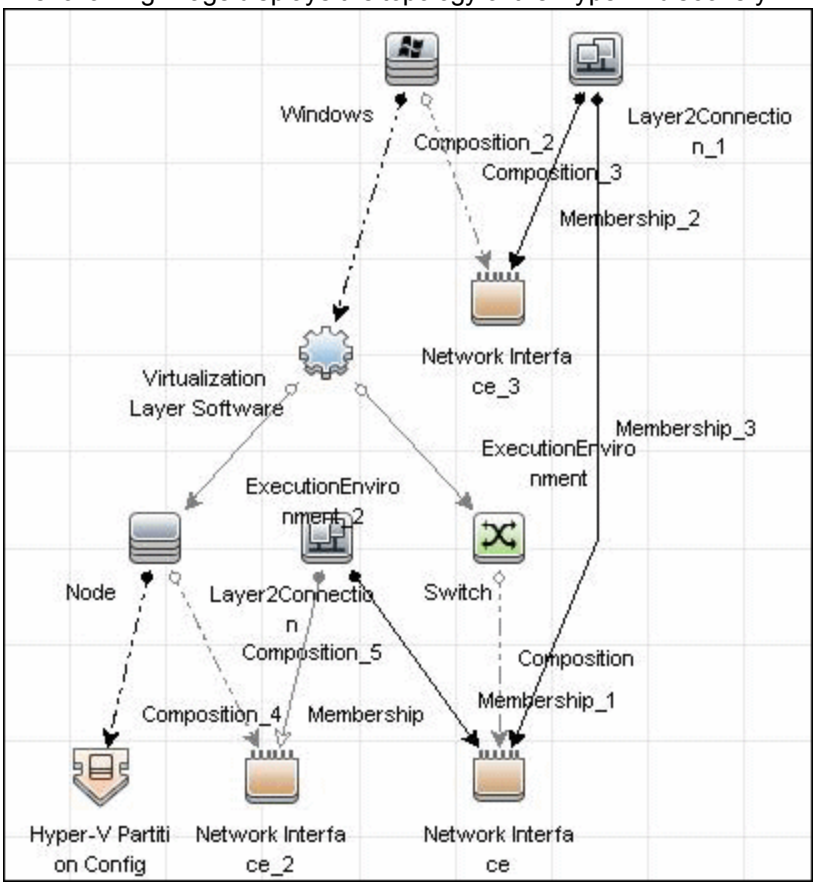
The **Hyper-V** package discovers the Hyper-V Aware Windows server through WMI and NTCMD. It discovers resource pools, virtual switches, virtual NICs, and virtual machines.

Supported Versions

The **Hyper-V** package supports Windows 2008 and Windows 2008 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" on page 82.

2. **Prerequisites - Verification**

   Verify that you can perform WMI queries in the **\\root\virtualization** 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 Resources and 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 Resources and 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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Discovery Mechanism

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 ("Retrieve the References from Virtual Machine Settings (VSSD) to Components" on previous page) 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 ("Retrieve the References from Virtual Machine Settings (VSSD) to Components" on previous page) 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.

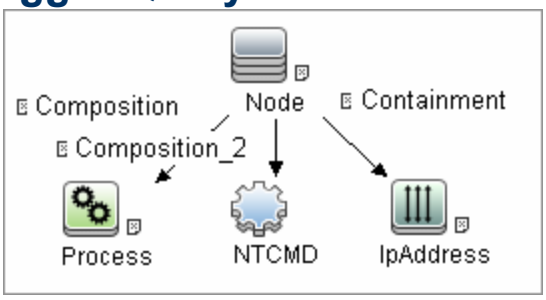
Retrieve the References from Ports on Virtual Switches to Interfaces

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.

The Hyper-V Topology by Shell Job

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

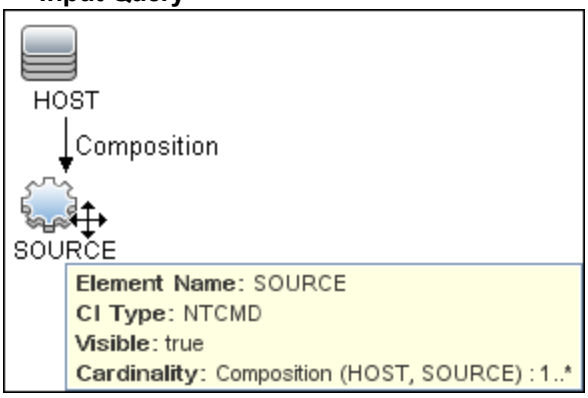
Trigger Query



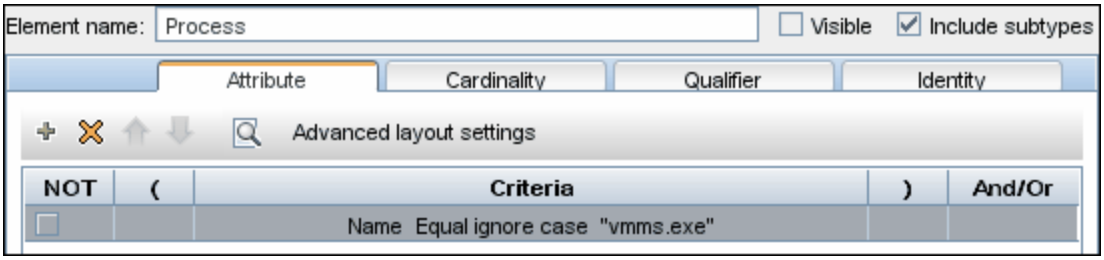
Adapter

This job uses the **hyperv_topology_by_shell** adapter.

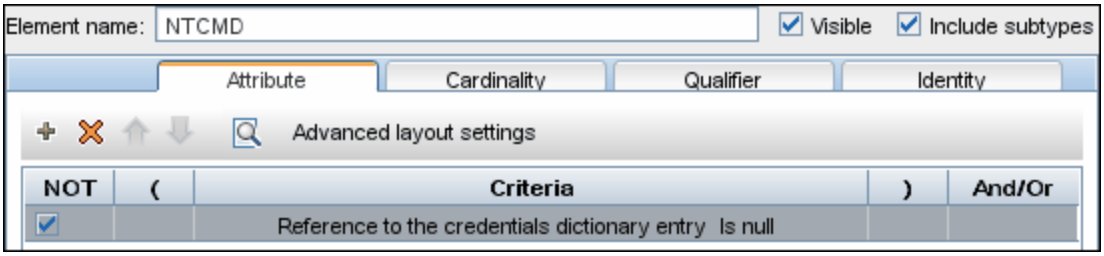
- **Input Query**



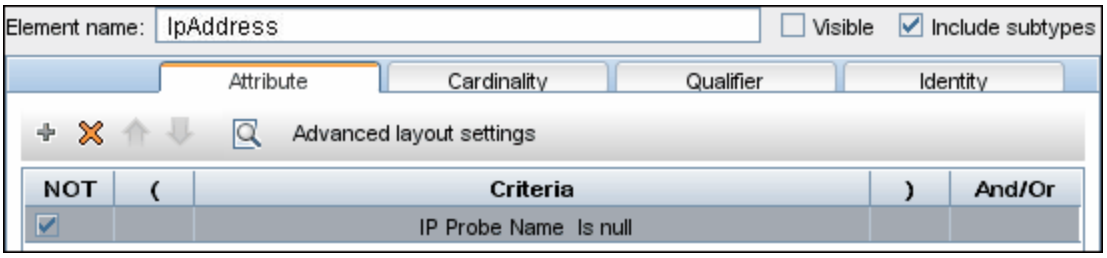
- **Process Element**



- **NTCMD Element**



- **IpAddress Element**

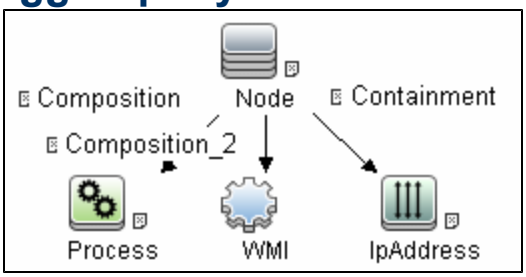


- **Discovered CITs**
  - **Composition**
  - **ExecutionEnvinroment**
  - **Hyper-V Partition Config**
  - **Interface**
  - **Layer2Connection**
  - **Membership**
  - **Node**
  - **Switch**
  - **Virtualization Layer Software**

The Hyper-V Topology by WMI Job

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

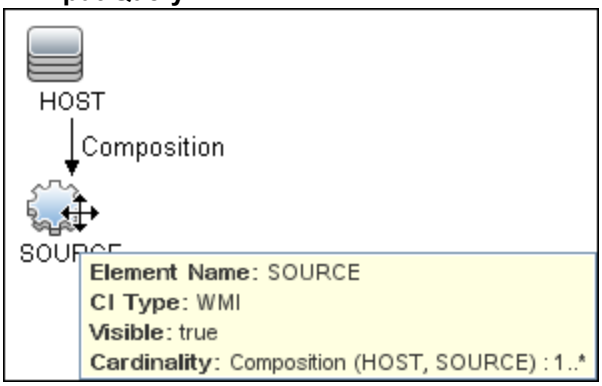
Trigger query



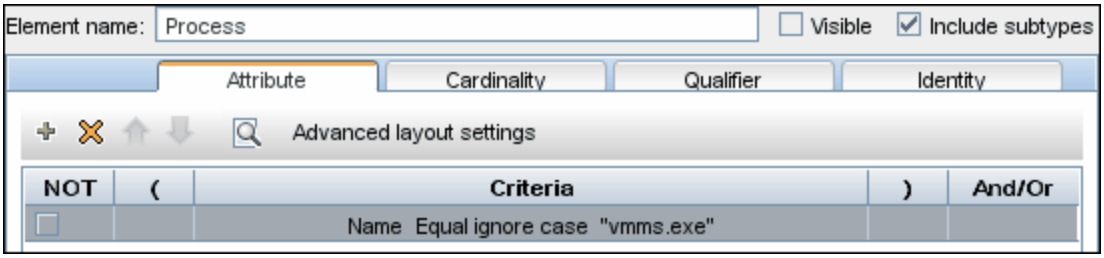
Adapter

This job uses the hyperv_topology_by_wmi adapter.

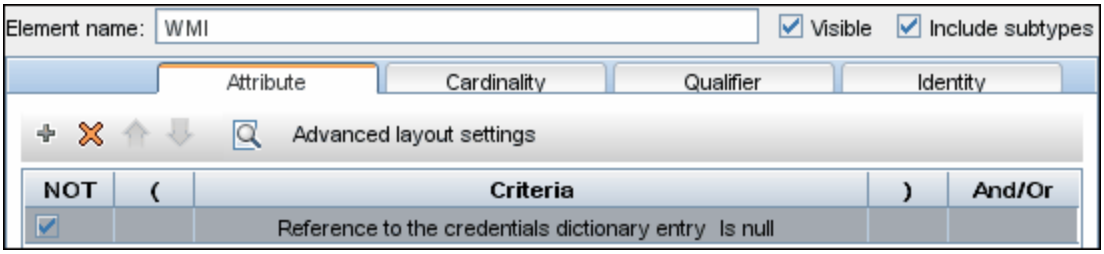
- **Input Query**



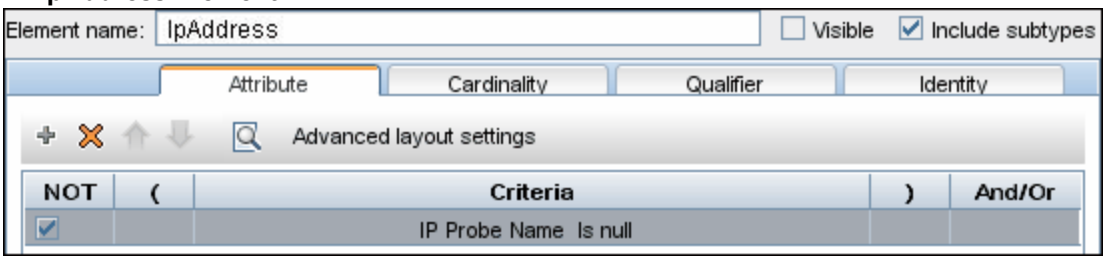
- **Process Element**



- **WMI Element**



- **IpAddress Element**



- **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
Module		Virtualization – Hyper-V (HyperV.xml)

Troubleshooting and Limitations

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

Chapter 76

IBM Hardware Management Console (HMC) Discovery

This chapter includes:

Overview

This document describes the usage and functionality of the IBM HMC discovery package.

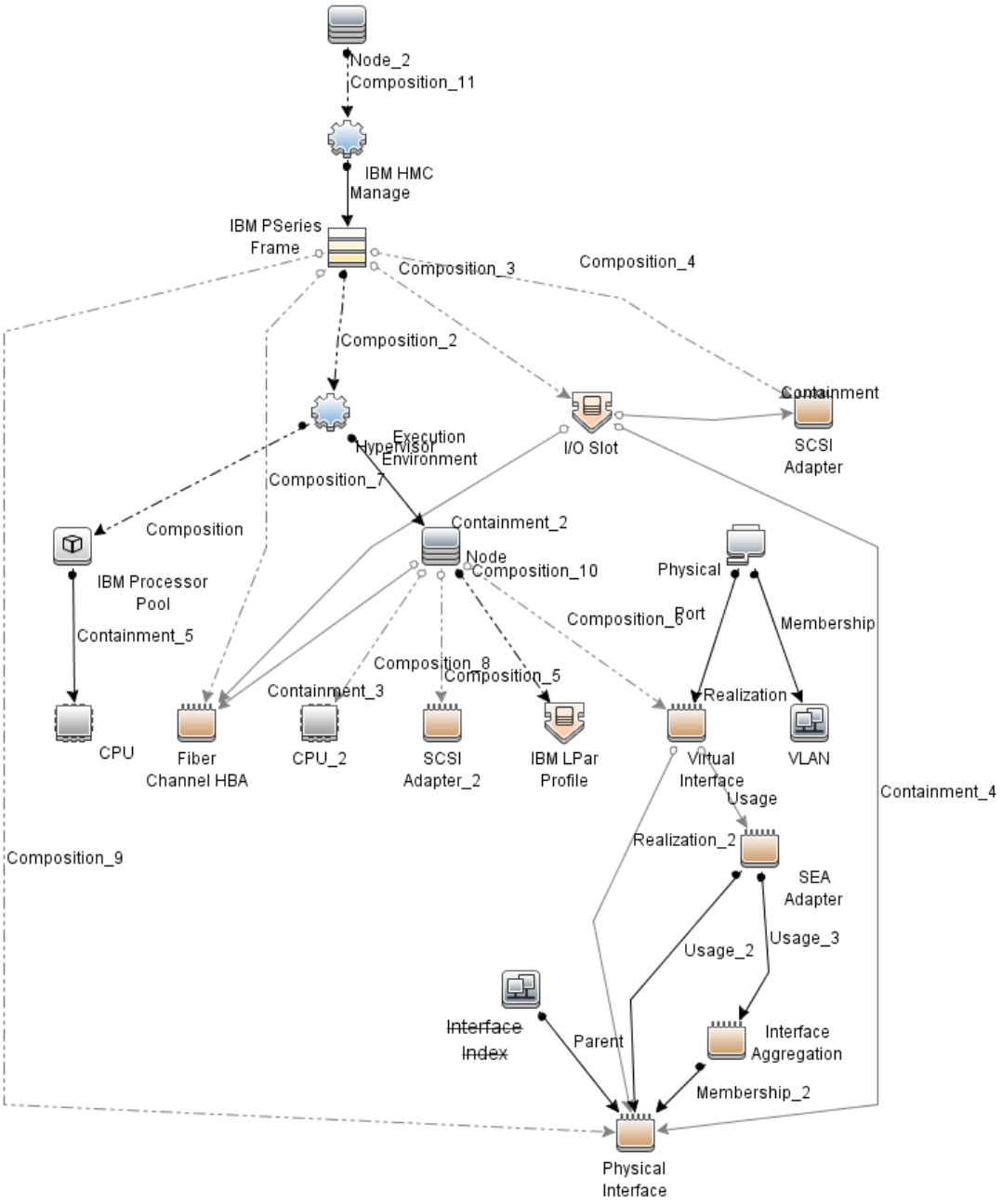
Hardware Management Console (HMC) is a technology invented by IBM for the purpose of providing a standard interface for configuring and operating partitioned (also known as an LPAR or virtualized system) and SMP systems such as IBM System I or IBM System p series.

Supported Versions

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

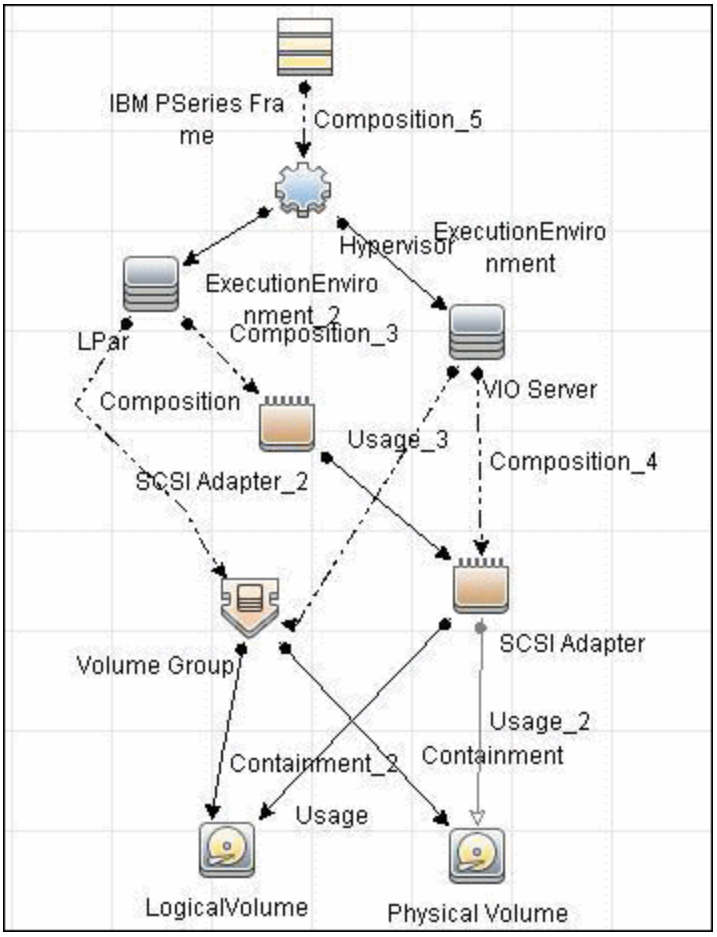
Topology

IBM HMC by Shell Topology



Note: For a list of discovered CITs, see "Discovered CITs" on page 1001.

IBM Storage Topology



Note: For a list of discovered CITs, see "Discovered CITs" on page 1004.

How to Discover IBM HMC

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" on page 82.

   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. For details about these commands, see:

- "IBM HMC Commands" on page 1006
- "VIO Server Side Commands" on page 1024
- "LPAR Side Commands" on page 1037

Command
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>
lslv
lslv -v <Logical Volume Name>
lsmap -all
lsmap -all -net

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

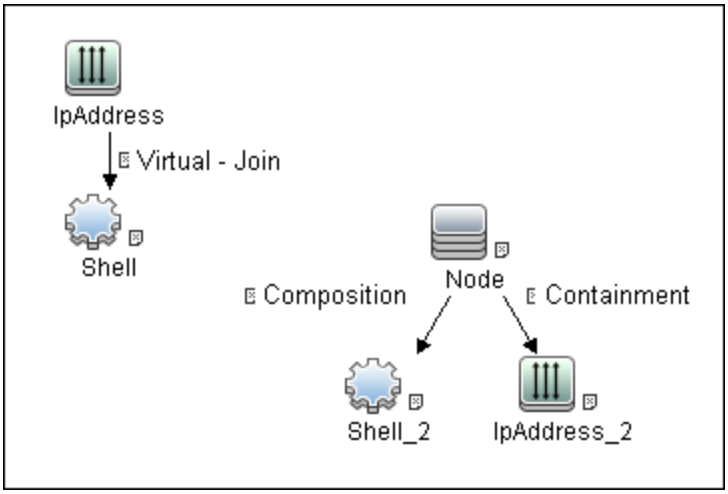
3. **Run the discovery**
   a. Run the **Range IPs by ICMP** job.
   b. Run the **Host Connection by Shell** job.
   c. Run the **IBM HMC by Shell** job.
   d. Run the **IBM LPar and VIO by Shell** job.

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

IBM HMC by Shell Job

This section includes:

Trigger Query

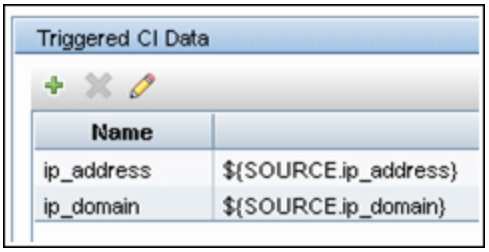


Adapter

This job uses the **IBM_HMC_SHELL_PATTERN** adapter.

- **Input Query**

- **Triggered CI Data**



- **Used Scripts**
  - **ibm_hmc_by_shell.py**
  - **storage_topology.py**
  - **ibm_hmc_lib.py**

Discovered CITs

- **Composition**
- **Containment**
- **Cpu**
- **ExecutionEnvironment**

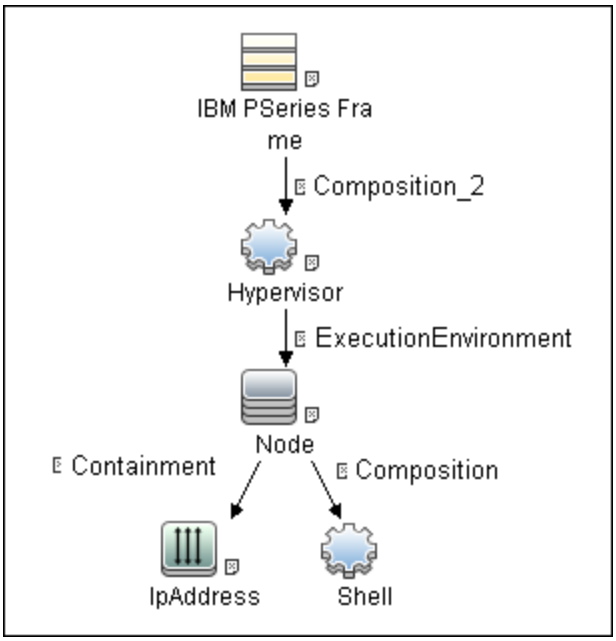
- **I/O Slot**
- **IBM Frame**
- **IBM HMC**
- **IBM LPar Profile**
- **IBM Processor Pool**
- **Interface**
- **IpAddress**
- **Manage**
- **Membership**
- **Node**
- **PhysicalPort**
- **Realization**
- **SCSI Adapter**
- **Shell**
- **Usage**
- **Virtualization Layer Software**
- **Vlan**

Note: To view the topology, see "IBM HMC by Shell Topology" on page 997.

IBM LPar and VIO by Shell Job

This section includes:

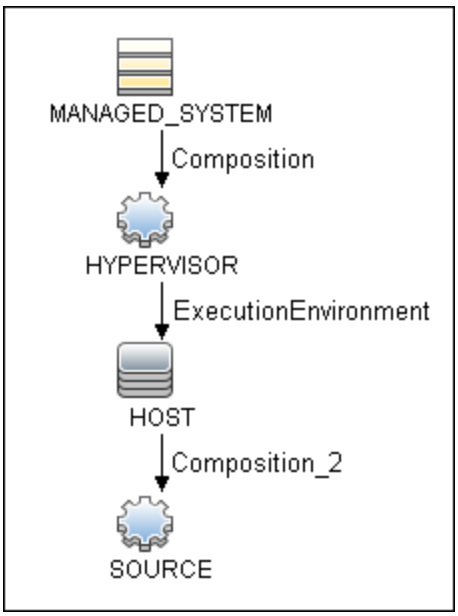
Trigger Query



Adapter

This job uses the **IBM_LPAR_VIO_BY_SHELL** adapter.

- **Input Query**



- **Triggered CI Data**

Name	
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**
- **LogicalVolume**
- **Membership**
- **Node**
- **Parent**
- **Physical Volume**
- **Realization**
- **SCSI Adapter**
- **SEA Adapter**
- **Usage**
- **Volume Group**

Note: To view the topology, see "Topology" on page 997.

IBM HMC 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
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
desired_num_huge_ pages	IBM LPAR Profile	LPAR Profile Desired Number of Huge Memory Pages

CMD Output Attribute	CI Name	CI Attribute
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
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

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

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

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 HMC by Shell	Job	Performs HMC based discovery
IBM LPAR and VIO Server Topology by Shell	Job	Performs LPAR and VIO Server side discovery
Virtualization - IBM HMC	Discovery Module	

Entity Name	Entity Type	Entity Description
IBM_HMC_BY_SHELL_PATTERN	Adapter	Adapter for the IBM HMC 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 HMC 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-HMC discovery.

- It is possible to configure the Partition Migration of an LPAR to the PFrame. This is supported only in P6, and is presently not supported by this solution.
- VIO Server on Linux OS is not supported.

Chapter 77

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

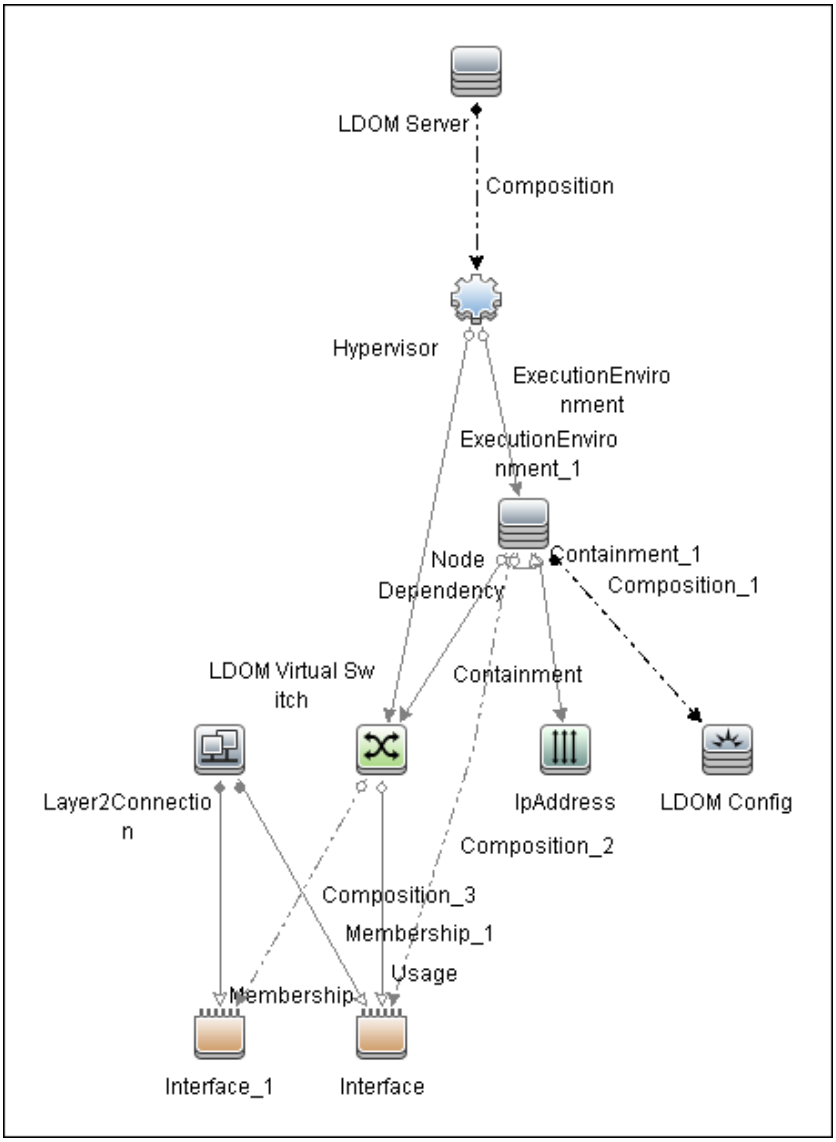
Oracle VM Server for SPARC Technology Discovery supports LDOM versions 1.0-1.3, and Oracle VM Server for SPARC versions 2.0-2.1.

Topology

This section displays the following topology maps:

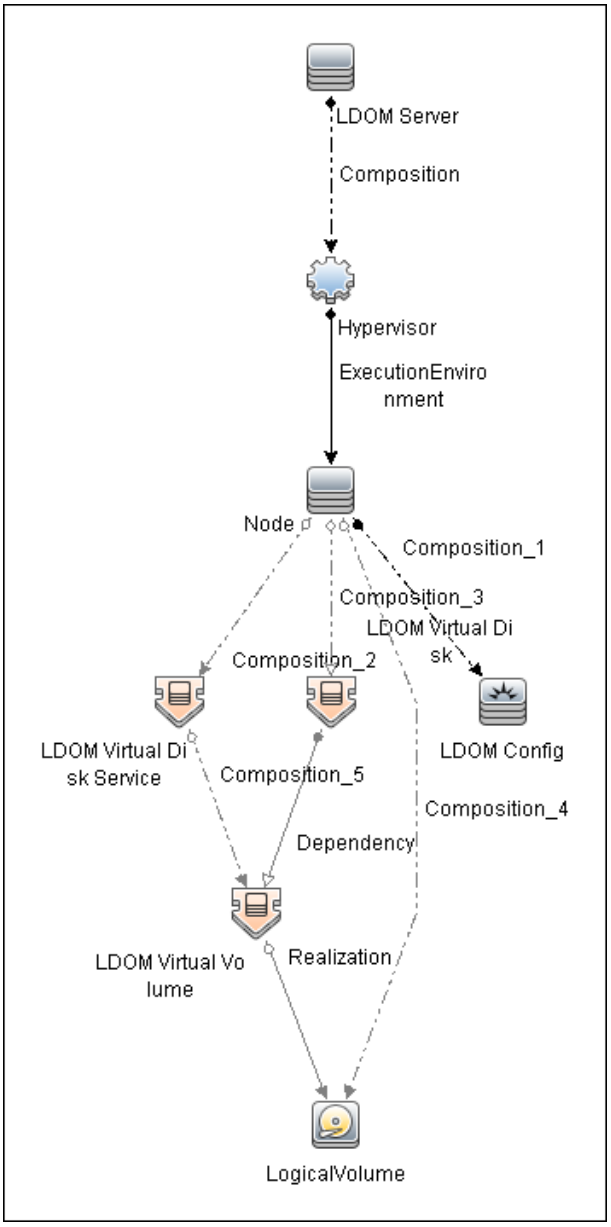
- "LDOM Networking and General Topology" below
- "LDOM Storage Topology" on next page

LDOM Networking and General Topology



Note: For a list of discovered CITs, see "Discovered CITs" on page 1046.

LDOM Storage Topology



Note: For a list of discovered CITs, see"Discovered CITs" on page 1046.

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" on page 82.

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 Resources and 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.

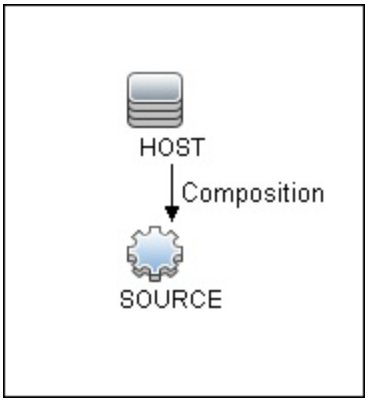
Oracle_VM_Server_for_SPARC_Technology_by_Shell Adapter

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

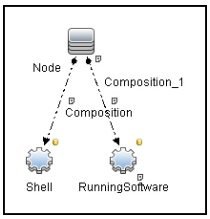
This section includes the following information:

Adapter

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 which 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 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 next page.
- **Get general networking configuration**

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

- netstat
- ifconfig
- dladm

For more information see "SunOs" (on page 1).

- **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 1053.

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/c6t600604800002901011775330323233 4d0s2|mpgroup=
|vol=L1_
2228|opts=|dev=/dev/dsk/c6t600604800002901011775330323232 38d0s2|mpgroup=
|vol=L1_
```

```
221C|opts=|dev=/dev/dsk/c6t600604800002901011775330323231 43d0s2|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
```

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 78

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.

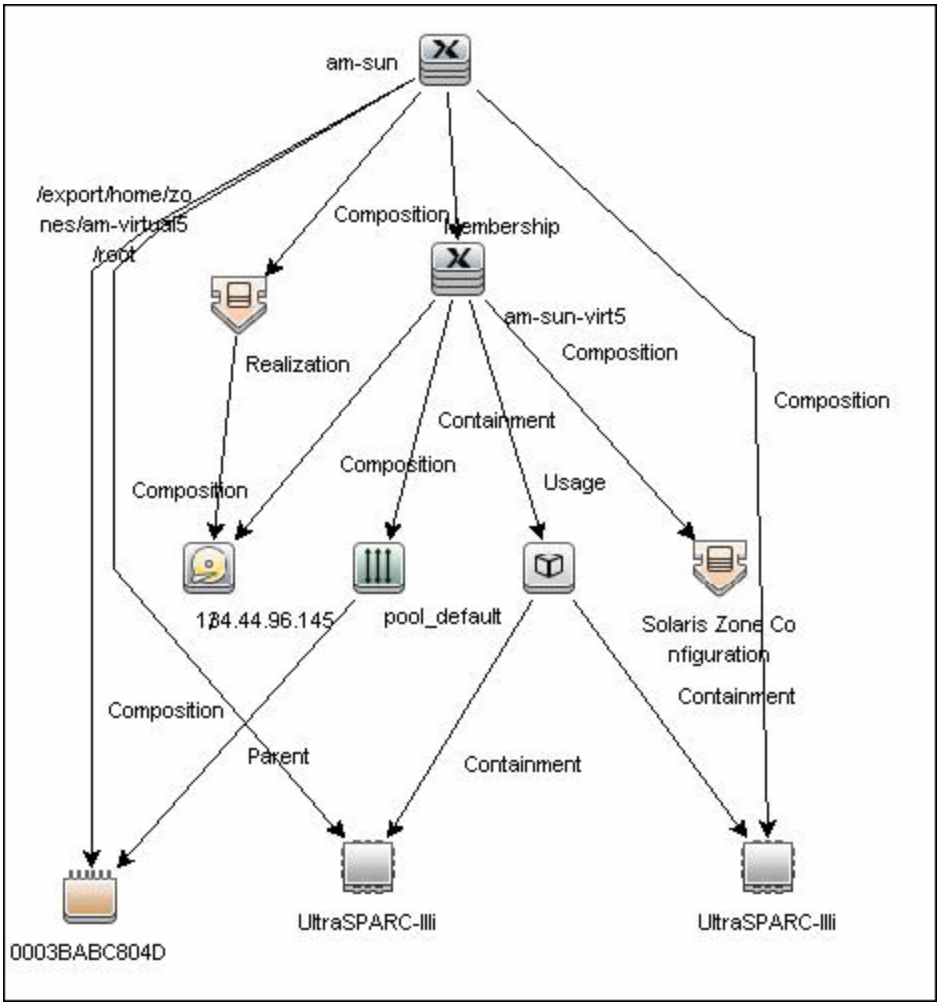
Supported Versions

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 "Discovered CITs" on page 1060.



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" on page 82.

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

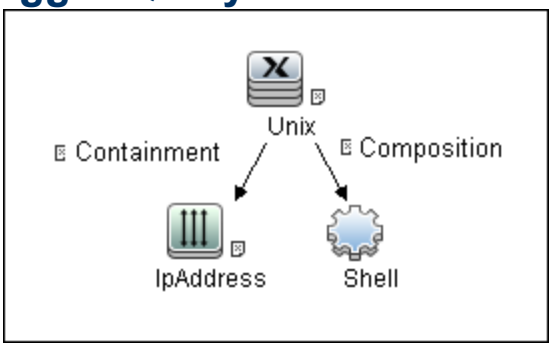
> **Note:** The zlogin command can be executed:
>
> i. With root user (the default value)
>
> ii. With a connection to the global zone user. You can configure this option with the discovery pattern parameter **zloginWithConnectedUser**.

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 "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

Solaris Zones by TTY Job

Trigger Query

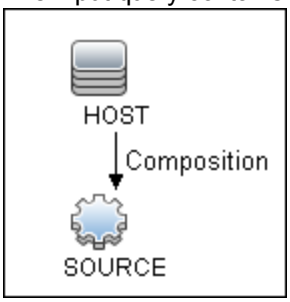


Adapter

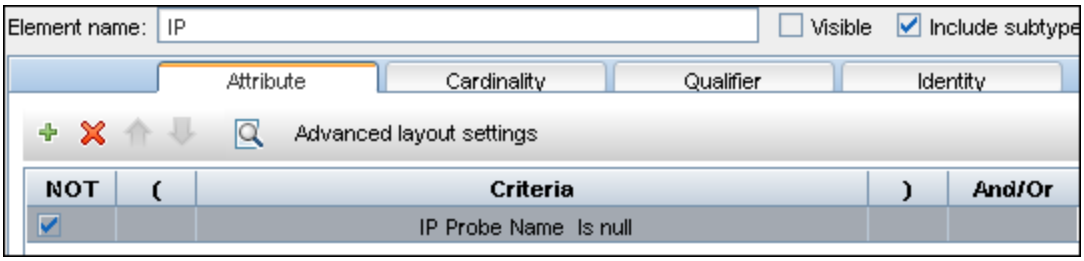
The Solaris Zones by TTY Job uses the **SolarisZone_Disc_By_TTY** adapter.

- **Input Query**

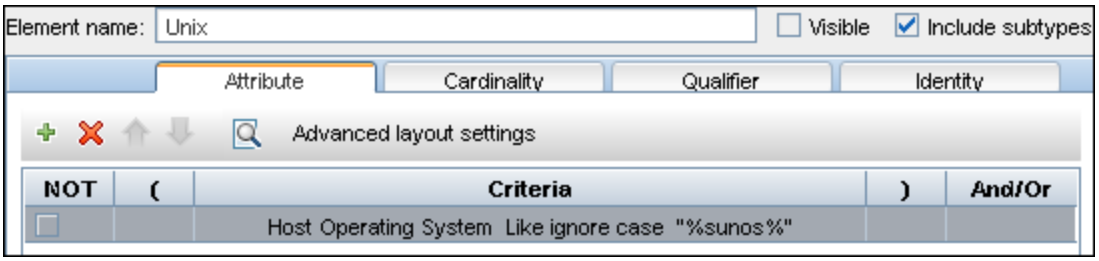
The Input query contains one Shell CI only:



- **IP Process**



- **UNIX Process**



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 1057.

Discovery Mechanism

This section includes the following commands:

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

This section describes troubleshooting and limitations fot Solaris Zones discovery.

- **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.
  - Add the **zlogin** command to the list of **sudo**-enabled commands.

Chapter 79

VMware Infrastructure Discovery

This chapter includes:

Supported Protocol Versions

With each new milestone release of VMware Infrastructure, new features and management entities are added to the product. As of now, the following versions of the protocols are supported by the servers:

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

Protocol versions supported by the server are tied to the version of the target servers. In general, servers are backwards compatible with regards supporting older versions of the protocol.

For example, ESX Server version 3.5 supports protocols 2.5 and 2.0.

It is not possible to retrieve information about features that were added later than the current version of the protocol. For example, while connected to ESX Server with protocol of version 2.0, it is not possible to retrieve information about DPM (Distributed Power Management) configuration because it was added only in version 2.5.

Currently DFM supports the above mentioned protocols and the discovery is adjusted according to the version of the protocol supported by the target server.

SSL Support

Web services use http transport which can also be transferred over SSL. The VMware Infrastructure Management (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 certificated. When connecting over SSL you should verify this certificate and accept it:

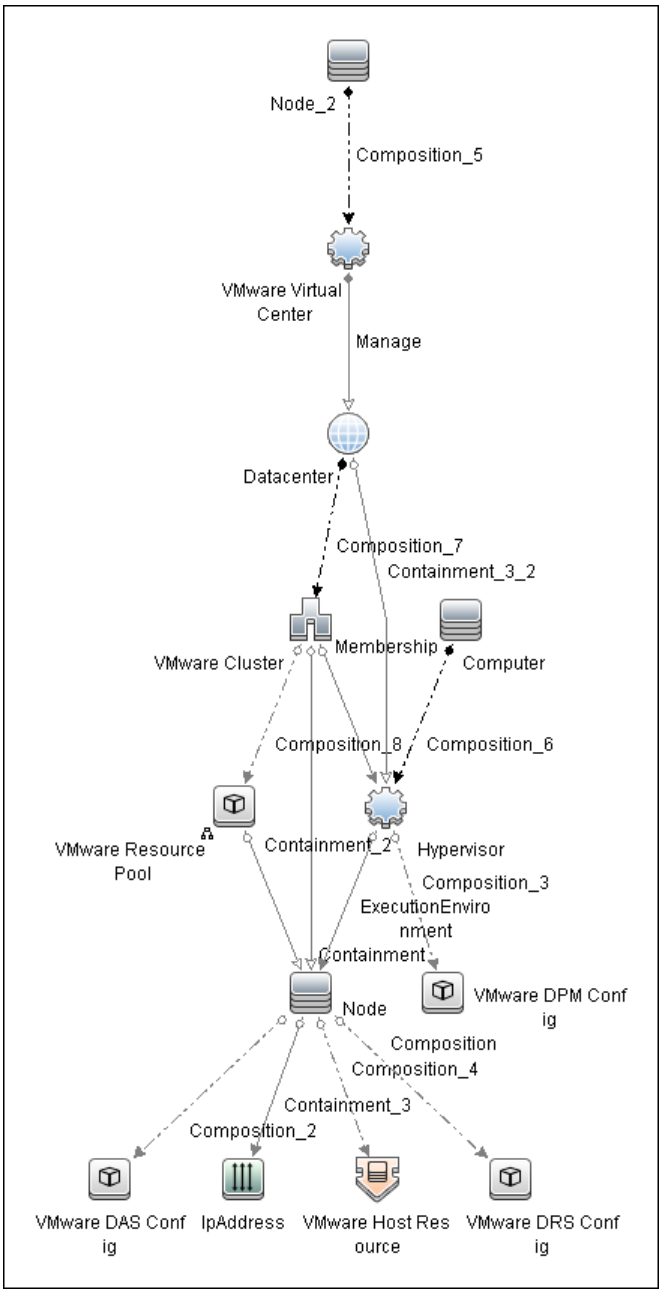
- Import all certificates from the server into a truststore and verify upon each connection while rejecting those that are not present in the set of trusted certificates (this is the secure method).
- Accept all certificates without verification (this is a less secure method).

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

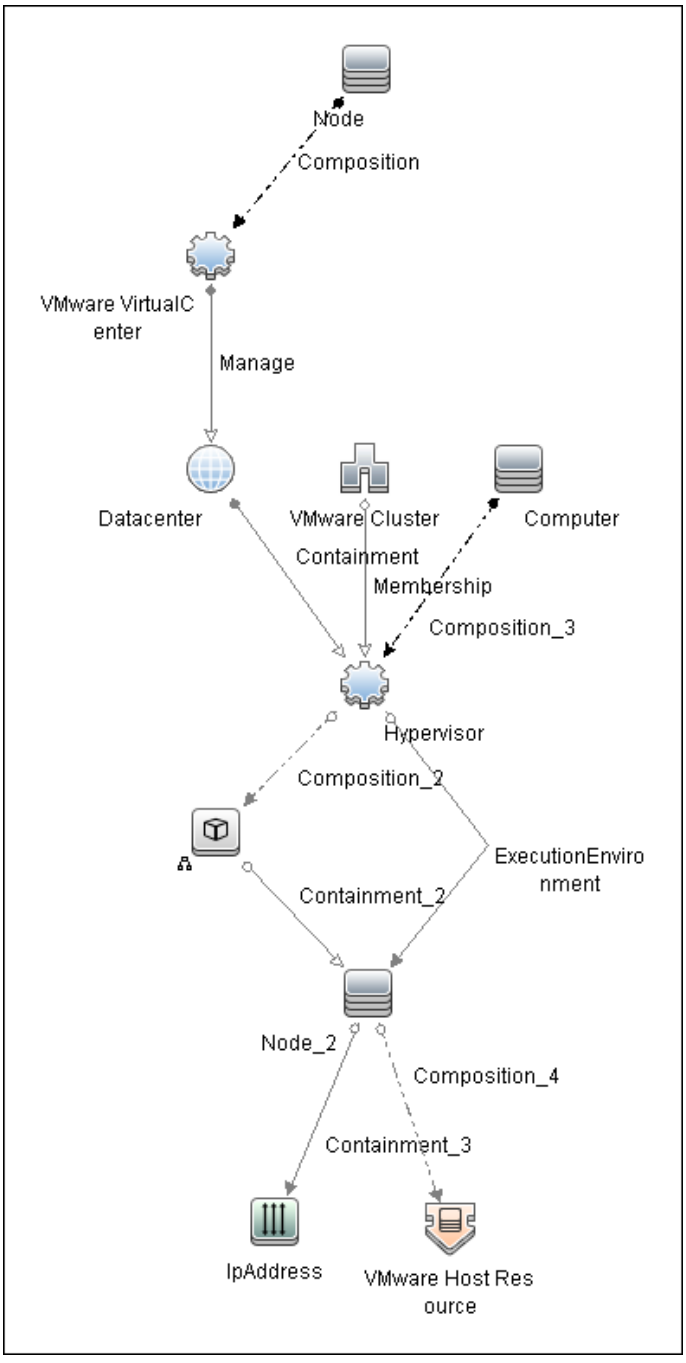
Topology

This section includes:

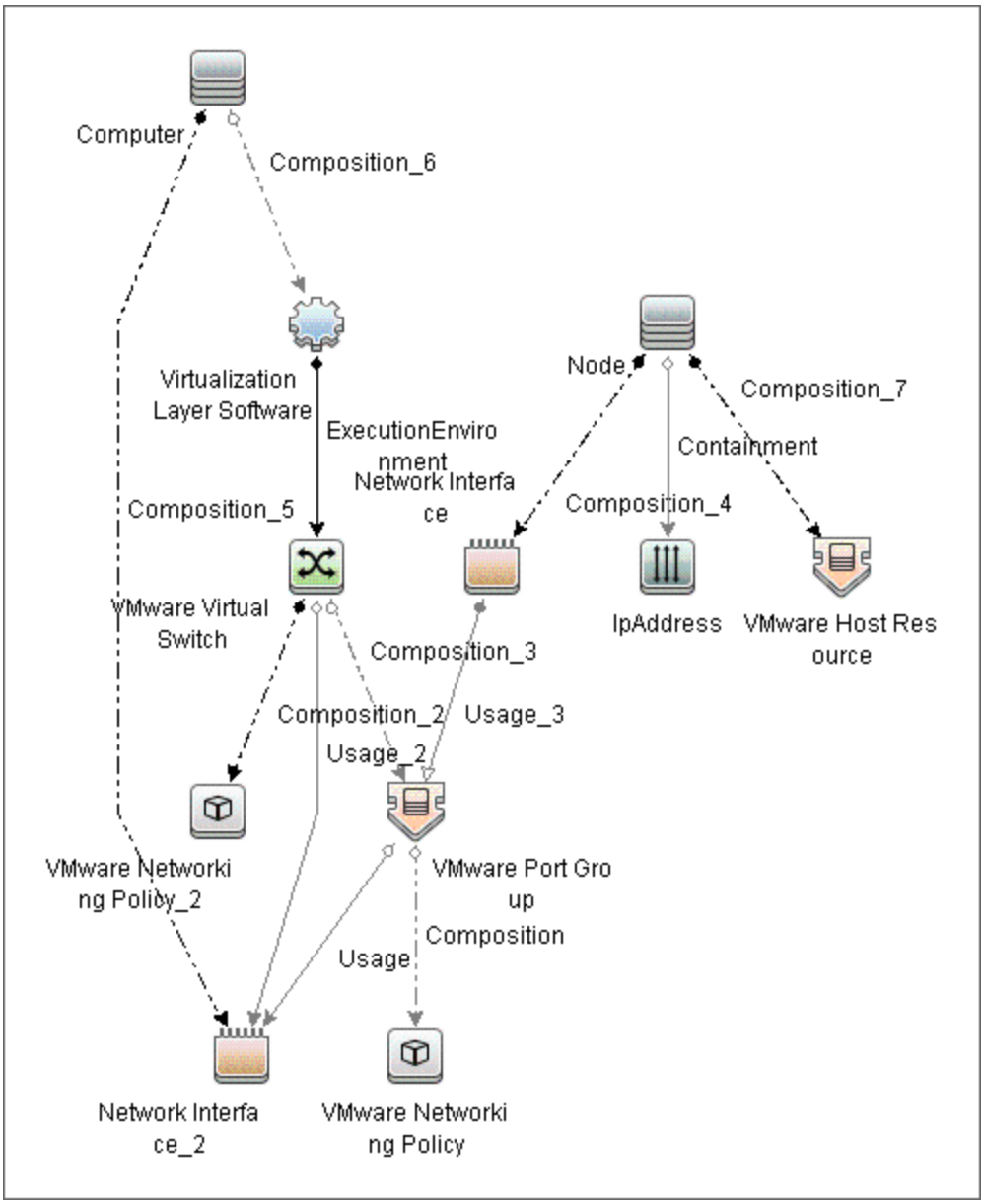
Virtual Topology View for Clusters



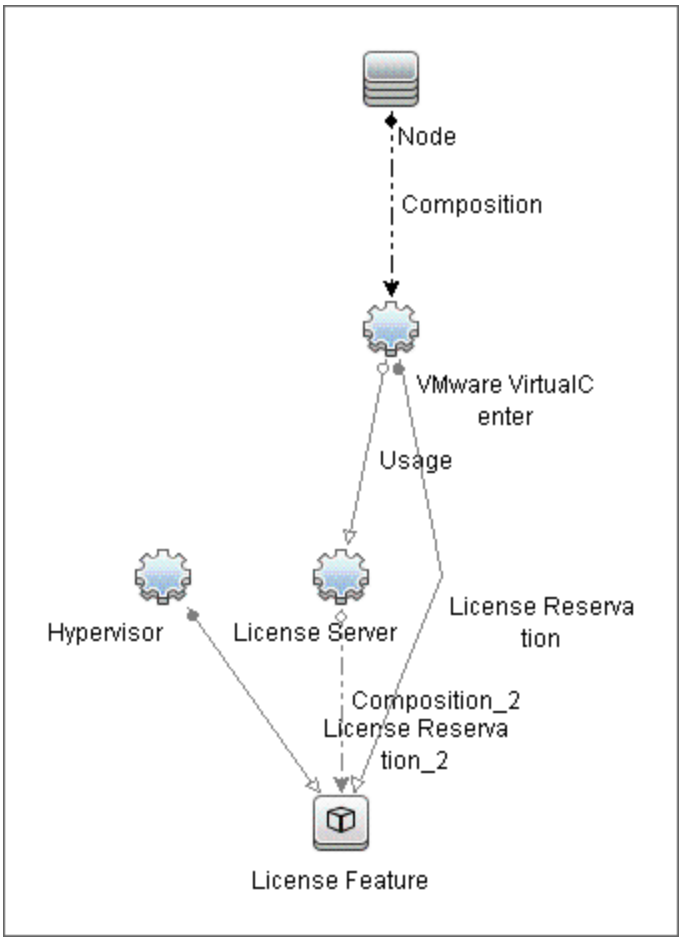
Virtual Topology View for Non-Clusters



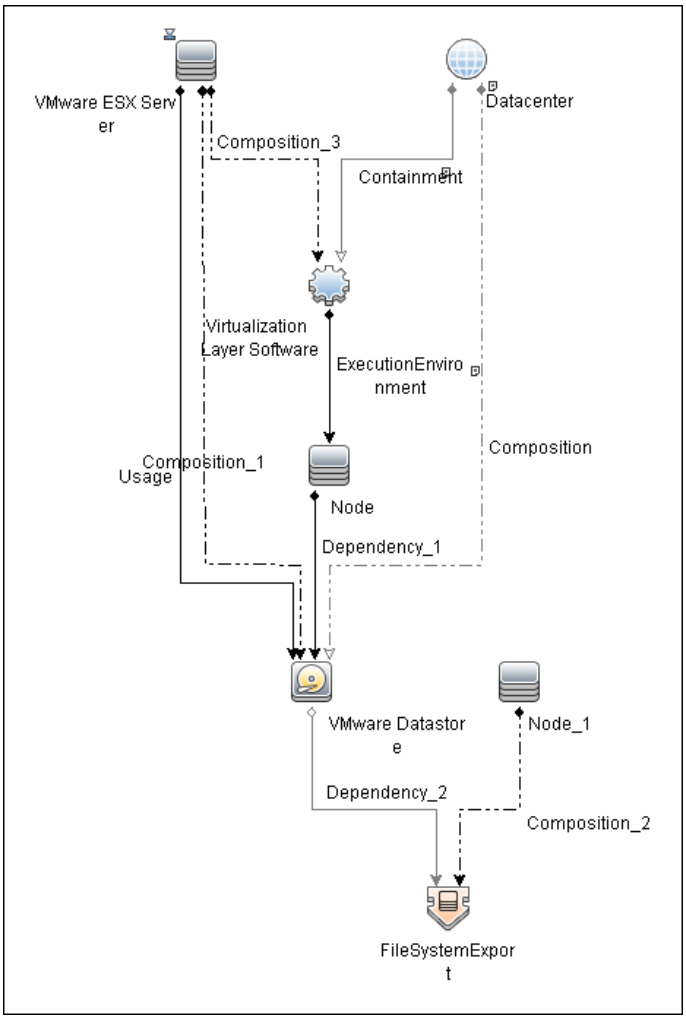
Virtual Topology View for Networking



Licensing Topology Map



Virtual Topology View for Storage



How to Discover VMware Infrastructure Topology

This task describes how to discover the VMware Infrastructure 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" on next page
- "Prerequisites – Add *.jar Files" on next page
- "Prerequisites – Set up VMware Infrastructure permissions" on next page

- "Run Host discovery" on next page
- "Run WMI discovery" on next page
- "Run Processes discovery" on next page
- "Run VMware Infrastructure discovery" on 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. The WMI protocol is required to discover the vCenter or VirtualCenter connectivity adapter.

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

   - The VMware Infrastructure Management (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" on page 82.

2. ## Prerequisites – Add *.jar Files

   To use the VMware Infrastructure Management protocol, add the following .jar files from the SDK to the Data Flow Probe:

   - **vim.jar**
   - **vim25.jar**

   These *.jar files are used without any modification together with the Axis engine. All protocol interactions are performed by working with objects from these *.jar files (instantiating objects, calling methods, getting result objects, and so on).

   > **Note:** These *.jar files are not included by default with DFM due to licensing issues.

   On each Probe running VMware discovery:

   a. Download the VMware Infrastructure SDK version 4.1 from the VMware support site (http://www.vmware.com/support/developer/vc-sdk).

   b. In the downloaded archive, in the **SDK\samples\Axis\java** folder, locate **vim.jar** and **vim25.jar**.

   c. Copy these .jar files to **C:\hp\UCMDB\DataFlowProbe\content\lib\vmware**.

   d. Restart the Probe.

3. ## 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.

The WMI protocol used in the vCenter or VirtualCenter connection adapter requires the following permissions:

- Users should be able to perform remote queries for the **root\default** namespace (**Remote Enable**, **Enable Account**, and **Execute Methods**); administrators usually have these permissions.

4. ## 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.

5. ## Run WMI discovery

   To connect to each potential vCenter or VirtualCenter server (this is not required for ESX), make the WMI connection available for the host by running the **Host Connection by WMI** job.

6. ## Run Processes discovery

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

7. ## 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 VirtualCenter Connection by WMI and VIM** job, followed by the **VMware VirtualCenter 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 four jobs cannot discover the VMware environment. You must, however, manually run this job, that is, you specify a URL (you need to know its format), you activate the job, and you choose the Data Flow Probe.
   - DFM models the Console Operating System (COS) as a Unix CI Type, and models the hardware running the ESX as a VMWare ESX Server CI Type. Once modeled,

these two CITs have the same or similar display names, but represent different entities, each one identified by its own set of unique properties.

For details about each job, see:

- "VMware VirtualCenter Connection by WMI and VIM Job" below
- "VMware VirtualCenter Topology by VIM Job" on page 1087
- "VMware ESX Connection by VIM Job" on page 1092
- "VMware ESX Topology by VIM Job" on page 1094

VMware VirtualCenter Connection by WMI and VIM Job

This job discovers vCenter or VirtualCenter Servers.

This section includes:

- "Discovery Mechanism" below
- "Trigger Query" on next page
- "Adapter" on next page
- "Discovered CITs" on page 1086
- "Troubleshooting" on page 1086

Discovery Mechanism

DFM runs the following processes:

- Runs through all defined credentials for the VMware Infrastructure Management (VIM) protocol.
- If the **Use SSL** parameter is set to **true**, the default prefix is HTTPS, otherwise the prefix is set to HTTP.
- If the user has entered a port number in the VIM protocol, this value is used for the port. If not, a WMI query is performed to extract the port number from the registry. DFM queries **HKLM\SOFTWARE\VMware, Inc.\VMware VirtualCenter** and searches for the **HttpsProxyPort** or **HttpProxyPort** attribute.
  - If the **HttpsProxyPort** attribute is found, DFM uses its value for the port and sets the prefix to HTTPS.
  - If the **HttpProxyPort** attribute is found, DFM uses its value for the port and sets the prefix to HTTP

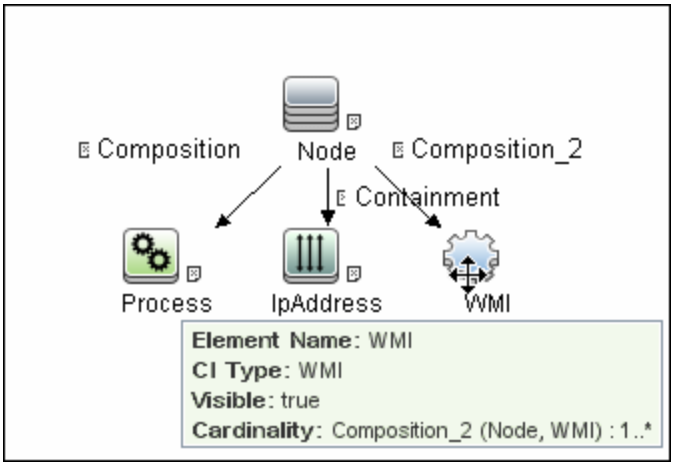
Note: DFM performs a search for the WMI port once only. The retrieved value is cached so that the same query does not need to be run for each VMware Infrastructure Management

(VIM) protocol entry.

- Once the port is found, DFM generates the connection URL as follows: **<prefix>://<ip_address>:<port>/sdk**.
- DFM creates a VMware Infrastructure Client, passes the user name and password from the current VMware Infrastructure Management (VIM) protocol, passes the generated URL, and performs a connection.
- If the connection is successful, DFM retrieves the product information and extracts the required values (these values are stored in the VMware VirtualCenter CI attributes). The values include build number, version, description, and so on.
- DFM uses the IP address to create a Host CI.
- DFM stores the generated URL used for this successful connection in the VirtualCenter CI's **connection_url** attribute.
- DFM stores the **credentialsId** of the current VIM protocol in the VirtualCenter CI's **credentialsId** attribute.
- If the connection is successful, DFM clears all errors and warnings that were generated in previous connection attempts and returns results.
- If the connection is unsuccessful, DFM continues with the next VIM protocol credentials entry, until all are tried.

Trigger Query

- **Trigger CI**: Host
- **Trigger query**:



Adapter

This job uses the **VMware_VirtualCenter_Connection_by_WMI_and_VIM** adapter.

- **Triggered CI Data**:

credentialsId	The credentials ID of the WMI agent CI.
ip_address	The IP address, taken from the WMI agent CI.
ip_addresses	List of all IPs connected to Host.

- **Adapter Parameters:** None.

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 (**Discovery Control Panel >Details** tab). For details, see "Discovery Permissions Window" in the *HP Universal CMDB Data Flow Management Guide*.

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

  ```
  Invalid user name or password
  ```

VMware VirtualCenter Topology by VIM Job

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

This section includes:

Discovery Mechanism

DFM performs the following processes:

1. DFM extracts the connection URL and the VIM protocol credentials ID by using the vCenter or VirtualCenter Trigger CI. DFM uses the credentials ID to retrieve the user name and password for the VIM protocol. DFM creates a VMware Infrastructure Client and connects to the server using these parameters.
2. DFM performs a query to retrieve information about Datacenters; the retrieved information is used to create Datacenter CIs.
3. DFM performs a query for the licensing information, including license availability and usage information, and information about license sources. The user used to retrieve availability information must have **Global.Licenses** permissions. If these permissions do not exist, DFM cannot add the **licenses_total** and l**icenses _available** attributes for each License Feature CI, and a warning is reported.
4. For each Datacenter, DFM performs a query to retrieve **ComputeResources** data. **`ComputeResource`** can represent either a single ESX server or a cluster (in which case it is called **`ClusterComputeResource`**). DFM does not map the **`ComputeResource`** resource itself to any CI (it is considered an abstract element of the hierarchy) but does use its properties.
5. For each **`ComputeResource`** resource that is a **`ClusterComputeResource`** resource, DFM treats the resource as a cluster and creates a Cluster CI. DFM performs an additional query to retrieve its attributes.
6. For each **`ComputeResource`** resource, DFM performs queries to retrieve:
   a. Information about its resource pools (the hierarchy of all the resource pools are retrieved in one query).
   b. Information about its ESX servers (all ESX servers are returned in one query; for a **`ComputeResource`** resource that is not a cluster, a single ESX is returned).
   c. Information about its VMs (all in one query).
7. For each ESX server, DFM discovers its licensing information. For details, see step 3 of

"Discovery Mechanism" on previous page.

8. When discovering VMs:

   a. DFM retrieves the host key for the **`Network Node`** CI, representing the guest OS, which can be the lowest MAC address, the IP address, or the UUID. If the host key cannot be found, DFM reports a warning in the communication log and the VM is skipped.

   b. DFM determines the power status of the VM: If it is powered-off, the **reportPoweredOffVms** parameter determines whether DFM skips the machine or includes it in the results. (You may not want to report a powered-off VM because the information it contains—for example, the IP address—may be outdated and may conflict with another VM that is powered-on.

   If **reportPoweredOffVms** is set to **false**, the powered-off VM is not reported.

   If **reportPoweredOffVms** is set to **true**, DFM tries to include the VM in the results (see the next step).

   c. All discovered VMs undergo a filtering mechanism. Currently filtering is performed by host keys. If there are two machines with the same host key, DFM reports only one, as follows:

   If both machines are powered-on, DFM reports the first that is found.

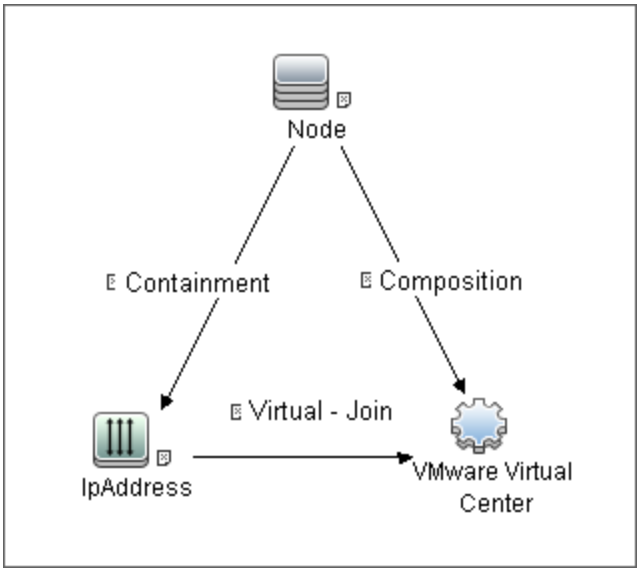
   If both machines are powered-off, DFM reports the first that is found.

   If the machines have different power states, DFM reports the powered-on machine.

9. All retrieved information is processed: DFM organizes the resource pools into a hierarchy and aligns each VM to its corresponding pool, then creates corresponding CIs and links, and returns the results.

Trigger Query

- **Trigger CI**. VirtualCenter.
- **Trigger TQL query**:



- **Node Conditions**. None.

Adapter

This job uses the **VMware_VirtualCenter_Topology_by_VIM** adapter.

- **Triggered CI Data**:

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

- **Adapter Parameters:**

reportPoweredOffVMs	Checks whether VMs that are powered off should be reported.

Discovered CITs

- **Composition**
- **Containment**
- **Cpu**
- **Datacenter**
- **Dependency**
- **ExecutionEnvironment**
- **FileSystemExport**
- **Interface**
- **IpAddress**
- **Licence Feature**
- **License Reservation**
- **License Server**
- **Manage**
- **Membership**
- **Node**
- **Usage**
- **VMware Cluster**
- **VMware DAS Config**
- **VMWare Datastore**

- **VMware DPM Config**
- **VMware DRS Config**
- **VMware ESX Server**
- **VMware Host Resource**
- **VMware Networking Policy**
- **VMware Port Group**
- **VMware Resource Pool**
- **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 and the CI is not reported:

  ```
  Cannot determine the IP or MAC address of virtual machine '<vm_name>
  ```

- **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.

VMware ESX Connection by VIM Job

This job discovers the connections to VMware ESX servers.

This section includes:

Discovery Mechanism

Data Flow Managementperforms 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.

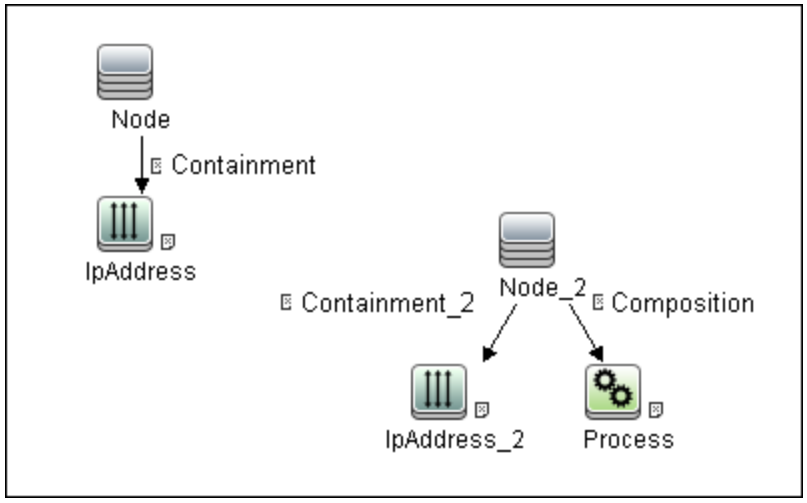
  In addition, DFM retrieves the UUID and name of the ESX server. ESX UUID is stored in the `host_key` attribute of the **VMware ESX Server** CI, which is a key attribute.

  The name of the ESX server is stored in the **data_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**: Host
- **Trigger query**:



Adapter

This job uses the **VMware_ESX_Connection_by_VIM** adapter.

- **Adapter parameters**. None.

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 VIM Job

This job connects to ESX servers and discovers their topology.

This section includes:

Discovery Mechanism

Data Flow Management performs the following procedure:

- DFM uses the connection URL (extracted from the ESX server attribute) and the user name and password (obtained by the **`credentialsId`** Trigger CI from the ESX server attribute) to connect to the server.
- DFM performs discovery of the ESX servers. DFM uses the same objects as the **`VMware VirtualCenter Topology by VIM`** job, so the flow is identical. (For details, see "VMware VirtualCenter Topology by VIM Job" on page 1087.)

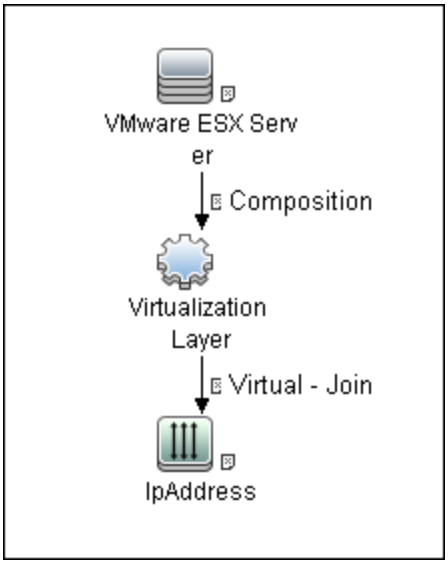
DFM discovers:

  - All resource pools of the server
  - All VMs of the server

- DFM performs discovery of the licensing information (as in the **`VMware VirtualCenter Topology by VIM`** job).
- DFM processes and returns results.

Trigger Query

- **Trigger CI**: Virtualization Layer Software
- **Trigger query and node conditions**:



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:**

reportPoweredOffVMs	Checks whether VMs that are powered off should be reported.

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
  ```

Chapter 80

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 VirtualCenter server exposes a management interface that can be used by DFM to:

- Connect to VirtualCenter 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 VMware Servers

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

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" on page 82.

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, refer to "Discovery Control Panel" in the *HP Universal CMDB Data*

Flow Management Guide.

a. Discover the VMware inventory structure, as described in "How to Discover VMware Infrastructure Topology" on page 1081.

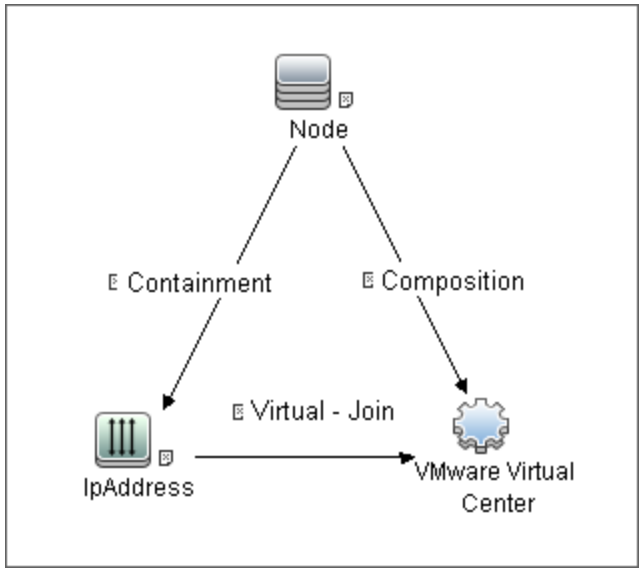
b. Activate the **VMware VMotion Monitor by VIM** job. The job includes the **VMware_ VMotion_discovery_by_VIM** adapter that listens for VM migration events collected by the VirtualCenter server.

VMware VMotion Monitor by VIM Job

This section includes:

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 VirtualCenter **application_ip**.
server_url	${SOURCE.connection_url}	The URL for connection, taken from the VirtualCenter **connection_url** attribute.

- **Adapter Parameters:**

connectionRetryNumber	The maximum number of times that DFM attempts to restore the connection. The default is **0** (zero), that is, the number of attempts is unlimited.
eventBasedDiscoveryEnabled	If this parameter is set to **true** (the default), every time the job is activated, it stays connected to the destination machine listening for VMotion events, until the job is stopped.
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. The default value is **24 hours**.

Discovered CITs

- **Composition**
- **Containment**
- **ExecutionEnvironment**
- **Interface**
- **IpAddress**
- **Node**
- **Usage**
- **VMware Host Resource**
- **VMware Port Group**
- **VMware Virtual Switch**
- **Virtualization Layer Software**

Chapter 81

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 (vim.jar,
  vim25.jar) are present in '<PROBE>\content\lib\vmware' folder.
  ```

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

  Solution. Copy the *.jar files to the Probe, as described in "How to Discover VMware Infrastructure Topology" on page 1081.

- **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 VirtualCenter.

  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 (either **VMware VirtualCenter Connection by WMI and VIM** or **VMware ESX Connection by VIM**) to update the `credentials ID attribute` of VirtualCenter or ESX server, and then run the topology job (**VMware VirtualCenter Topology by VIM** or **VMware ESX Topology by VIM)**.

Limitations

- If a VM's **host_key** attribute cannot be found, the VM is ignored and is not reported to HP Universal CMDB.
- 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 82

Xen Discovery

This chapter includes:

Overview

The Xen hypervisor, the open source industry standard for virtualization, virtualizes x86, x86_64, IA64, ARM, and other CPU architectures. It supports guest operating systems including Windows, Linux, Solaris, and various versions of the BSD operating systems.

Supported Versions

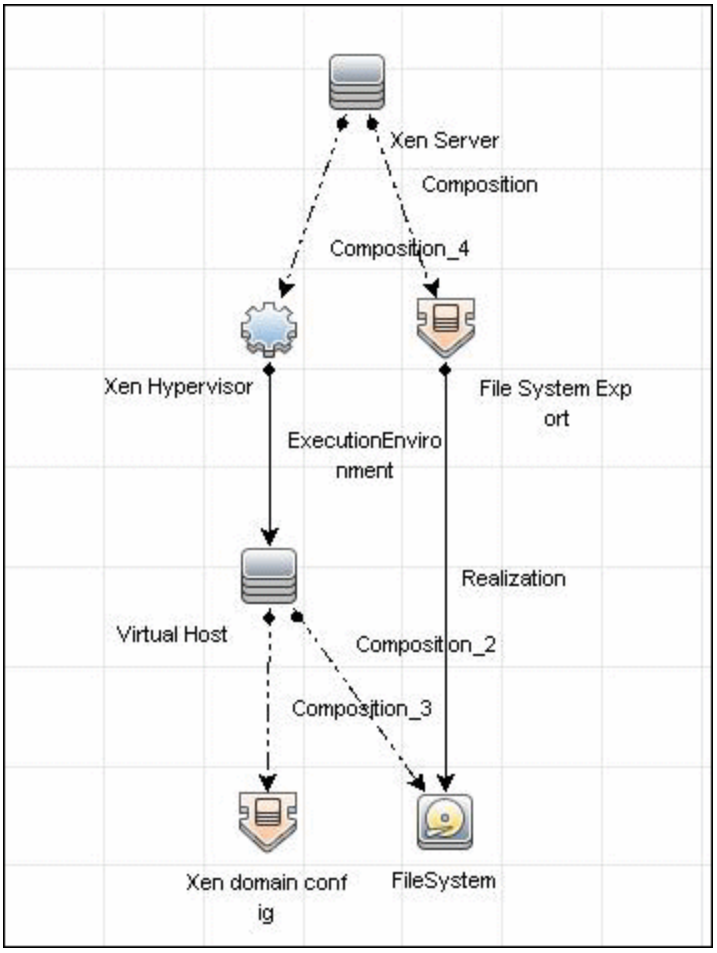
This discovery solution supports Xen 3.x.

Topology

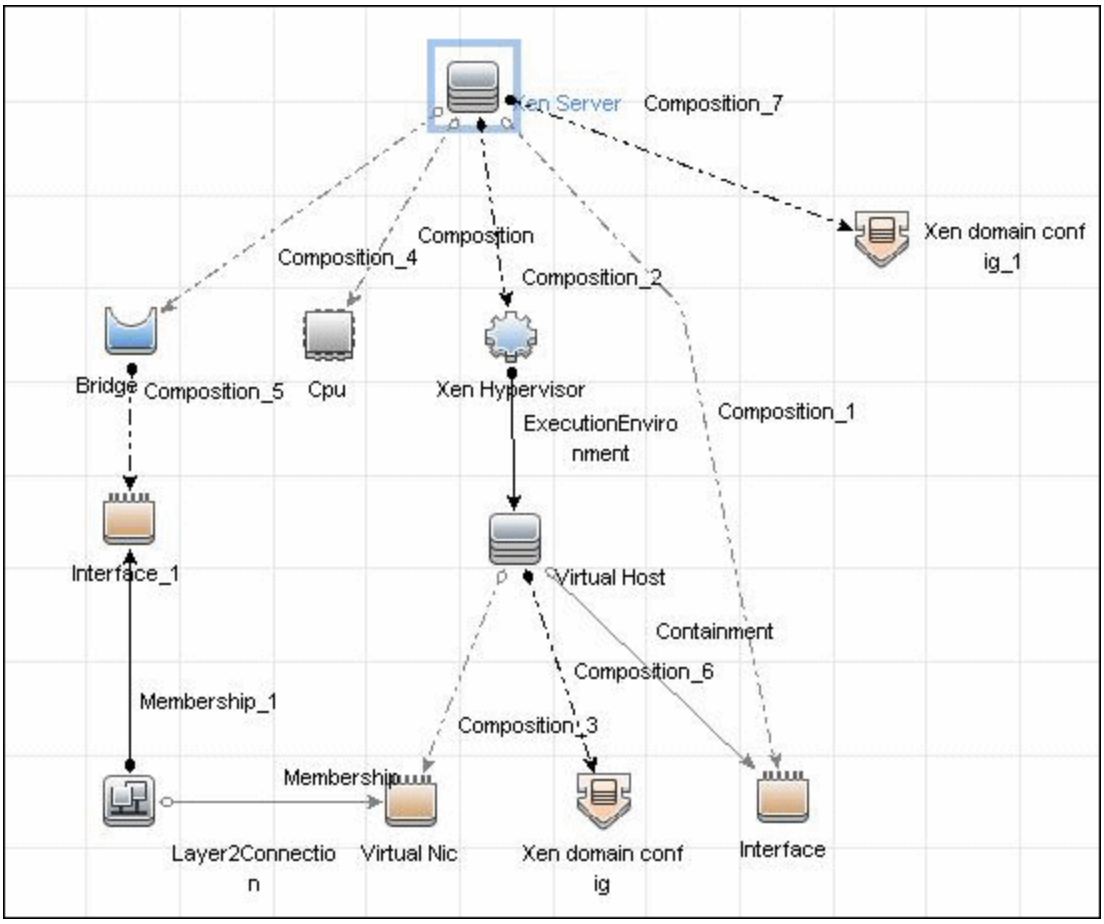
The following images display the topology of the Xen discovery jobs.

Note: For a list of discovered CITs, see "Discovered CITs" on page 1119.

Xen Storage Topology



Xen Topology



How to Discover Xen

This task includes the following steps:

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

   For credential information, see "Supported Protocols" on page 82.

2. **Prerequisites - Set up Xen parameters**

   a. Add SSH credentials for the Xen server.

   b. If the **xm** command is not located in a standard path (for example, **/bin**, **/sbin**, **/usr/bin**, or **/usr/sbin**), you must either add the path to **xm** in the **PATH** OS environment variable, or specify the path to it in the job property in the XEN by TTY job parameters tab.

   c. 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`

   d. Make sure that the discovery user has permissions to connect to the Xen server and to run the following commands:

      - xm info
      - xm list
      - xm list -l <domain_name>
      - brctl show
      - ifconfig -a

      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 **Xen Topology by TTY** job.

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

Xen Topology by TTY Discovery Job

This section includes:

Discovery Mechanism

This section includes the following commands:

Map Output to CI Attributes – for Xen Hypervisor and Hardware Resources

Command	xm info
Output	host : VMAMQA348.devlab.ad release : 2.6.18-194.3.1.el5xen version : #1 SMP Sun May 2 04:26:43 EDT 2010 machine : x86_64 nr_cpus : 2 nr_nodes : 1 sockets_per_node : 2 cores_per_socket : 1 threads_per_core : 1 cpu_mhz : 2932 hw_caps : 0febfbff:28100800:00000000 :00000140:80982201:00000000:0 0000001 total_memory : 8191 free_memory : 5442 node_to_cpu : node0:0-1 xen_major : 3 xen_minor : 1 xen_extra : .2-194.3.1.el5 xen_caps : xen-3.0-x86_64 xen-3.0-x86_32p xen_pagesize : 4096 platform_params : virt_start=0xffff800000000000 xen_changeset : unavailable cc_compiler : gcc version 4.1.2 20080704 Red Hat 4.1.2-48) cc_compile_by : mockbuild cc_compile_domain : redhat.com cc_compile_date : Sun May 2 04:16:18 EDT 2010 xend_config_format : 2

Output of this command is used to populate the attributes of the CIs:

CMD Output Attribute	CI Name	CI Attribute Display Name
xen_major +"." xen_minor	Hypervisor	Application version (application_version_ number)
xen_major +"." +xen_minor+ xen_ extra	Hypervisor	Application Version Description
nr_cpus	Xen domain config	Xen Number of Processors
sockets_per_node	Xen domain config	Xen Sockets number
threads_per_core	Xen domain config	Xen Threads per Core
total_memory	Xen domain config	Xen Total memory
free_memory	Xen domain config	Xen Free Memory

Use Output to Create List of Domains

Command	`xm list`

Output	`Name ID Mem(MiB) VCPUs State Time(s)`
	`Domain-0 0 2048 2 r----- 15771.6`
	`fedora12_64 9 512 1 -b---- 1272.4`

Mapping	The output creates a list of Domains running on the particular Xen server.

Map Output to CI Attributes – for Domain Configuration Information

Command	xm list -l fedora12_64 xm list -l <domain_name>
Output	(domain (domid 9) (uuid d2ea72a3-7d27-933e-021e-2d7ec1f05081) (vcpus 1) (cpu_cap 0) (cpu_weight 256.0) (memory 512) (shadow_memory 0) (maxmem 512) (bootloader /usr/bin/pygrub) (features) (name fedora12_64) (on_poweroff destroy) (on_reboot restart) (on_crash restart) (image (linux (ramdisk /var/lib/xen/boot_ramdisk.pkJA8q) (kernel /var/lib/xen/boot_kernel.B7TO_v) (args 'ro root=/dev/mapper/VolGroup-lv_root LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYTABLE=us

Command	xm list -l fedora12_64 xm list -l <domain_name>

Output (cont'd)	<pre>console=hvc0 rhgb quiet'
	)
	)
	)
	(cpus ())
	(device
	(vif
	(backend 0)
	(script vif-bridge)
	(bridge virbr0)
	(mac 00:16:36:61:12:c6)
	)
	)
	(device
	(tap
	(backend 0)
	(dev xvda:disk)
	(uname tap:aio:/mnt/vmimages/fedora12_64.img)
	(mode w)
	)
	)
	(state -b----)
	(shutdown_reason poweroff)
	(cpu_time 1272.36904274)
	(online_vcpus 1)
	(up_time 961277.138582)
	(start_time 1277970939.8)
	(store_mfn 2287142)
	(console_mfn 2287141)
	)</pre>

Output of this command is used to populate the attributes of the CIs:

CMD Output Attribute	CI Name	CI Attribute Display Name
domid	Xen domain config	Xen Domain Id
uuid	Host	host BIOS UUID
vcpus	Xen domain config	Xen virtual CPU Count
memory	Xen domain config	Xen Domain Memory
name	Xen domain config	Xen Domain Name
on_poweroff	Xen domain config	Xen Domain on Power Off Action

CMD Output Attribute	CI Name	CI Attribute Display Name
on_reboot	Xen domain config	Xen Domain on Restart Action
on_crash	Xen domain config	Xen Domain on Crash Action
state	Xen domain config	Xen Domain State
bridge	Bridge	Name
uname tap:aio:	Network Share	Name
mac	Network Interface	Interface MAC Address

Use Output to Retrieve Relationship Between Bridge and Bridged

Command	`brctl show`

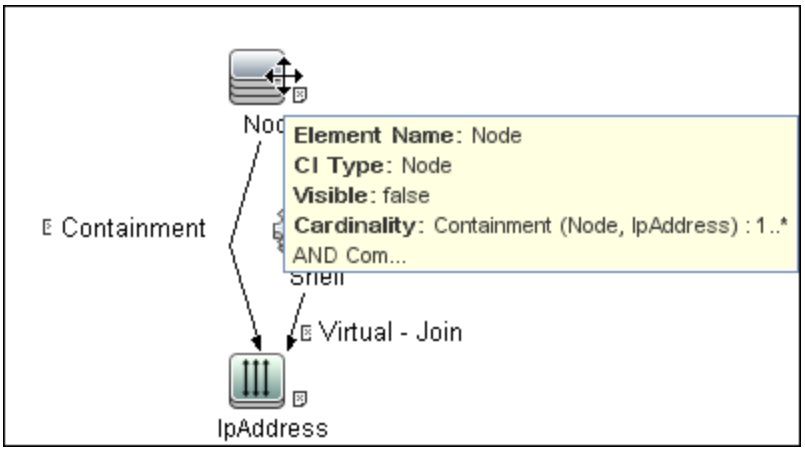
Output	`bridge name bridge id STP enabled interfaces`
	`br0 8000.0050569f684a no eth0`
	`peth0`
	`virbr0 8000.feffffffffff yes vif9.0`

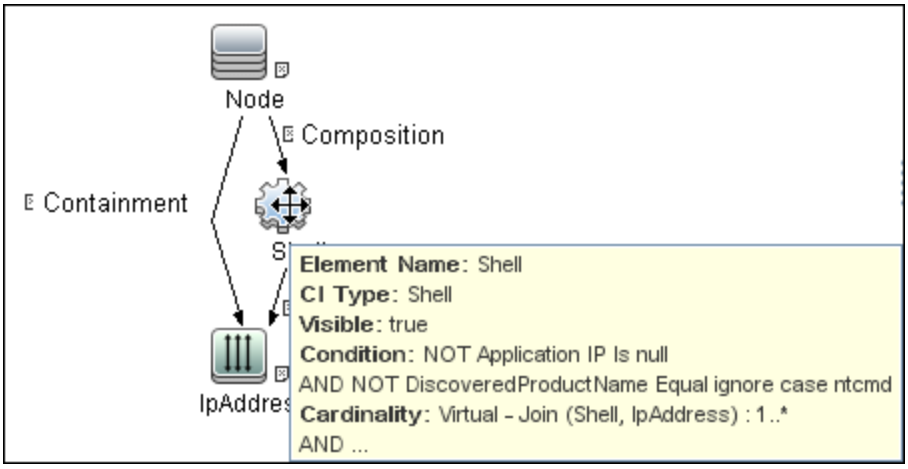
Mapping	

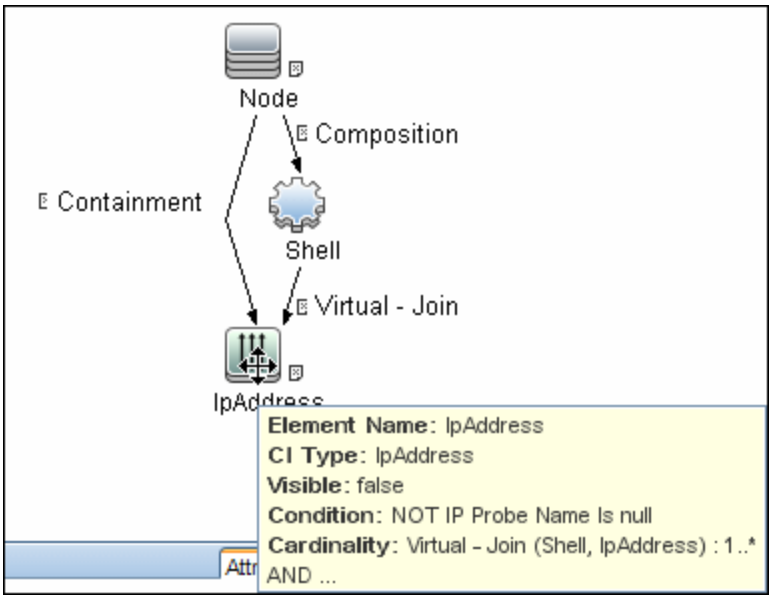
Output of this command is used to populate the attributes of the CIs:

CMD Output Attribute	CI Name	CI Attribute Display Name
bridge name	Bridge	Name
bridge id	Bridge	Bridge Base MAC Address
interfaces	Network Interface	Name

Trigger Queries

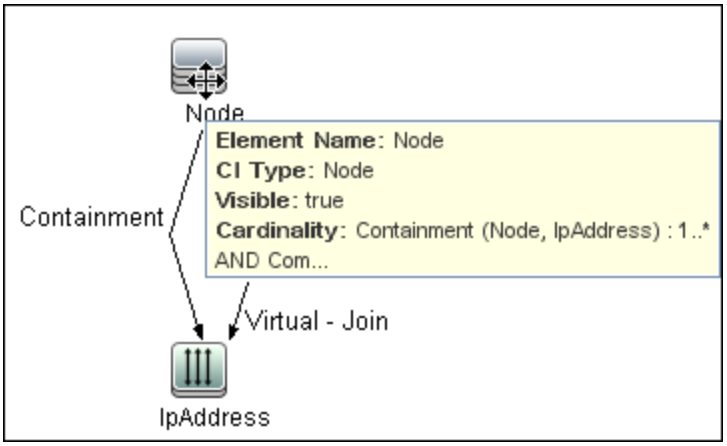


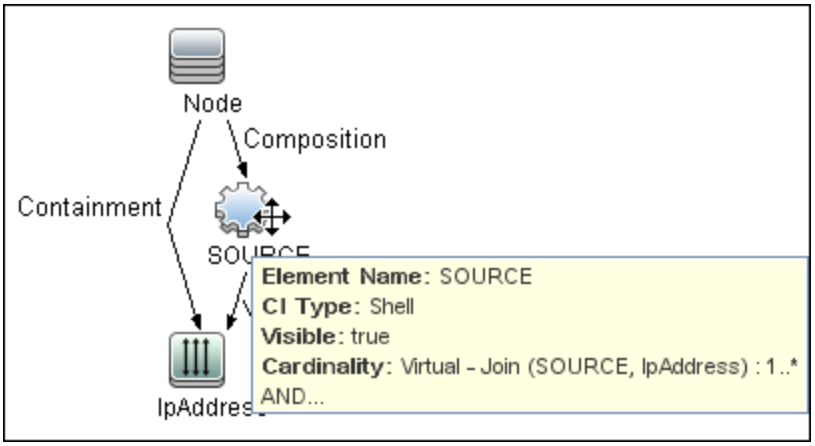


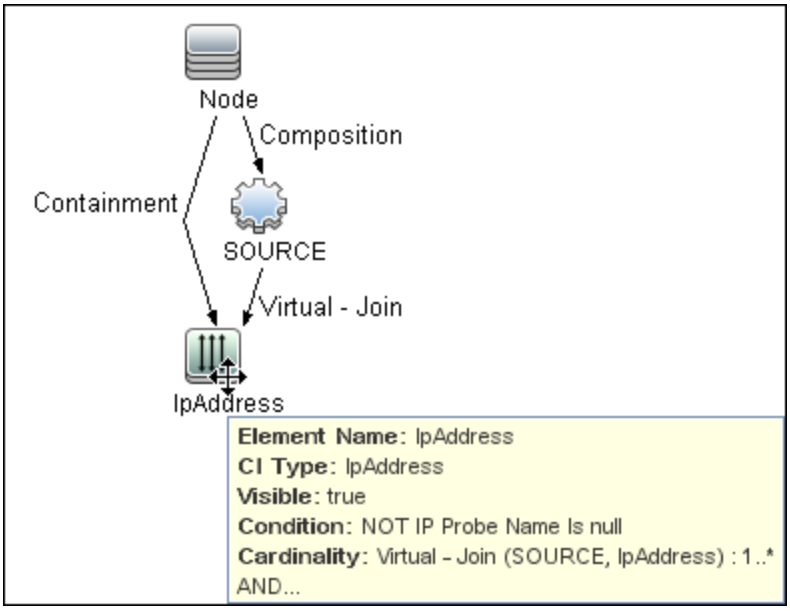


Adapter

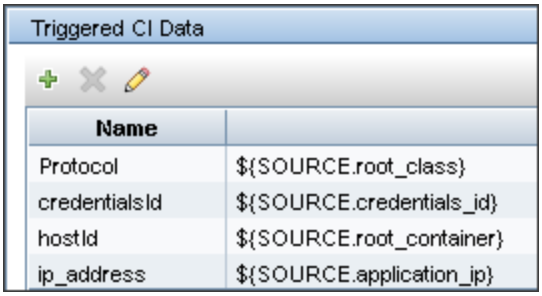
- **Input Queries**







- **Triggered CI Data**



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

- **Used Script**
  - **xen_by_tty.py**
- **Xen_by_TTY Adapter Parameters**
  - **xm_path.** Path to the xm management utility

Created/Changed Entities

Entity Name	Entity Type	Entity Description
xen_domain_config.xml	CIT	Domain configuration and parameters
Xen Topology by TTY.xml	Job	Main job
Virtualization - Xen.xml	Module	Discovery module
Xen_by_TTY.xml	Adapter	Discovery adapter
xen_by_tty.py	script	Discovery Jython script
xen_unix_with_shell.xml	query	Trigger query
Xen Topology.xml	View	View of the discovered topology
Xen Storage Topology.xml	View	View of the storage topology
containment.host.interface.xml	Valid link	
composition.bridge.interface.xml	Valid link	

Discovered CITs

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

Note: To view the topology, see "Topology" on page 1106.

Part XII: Web Servers

Chapter 83

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

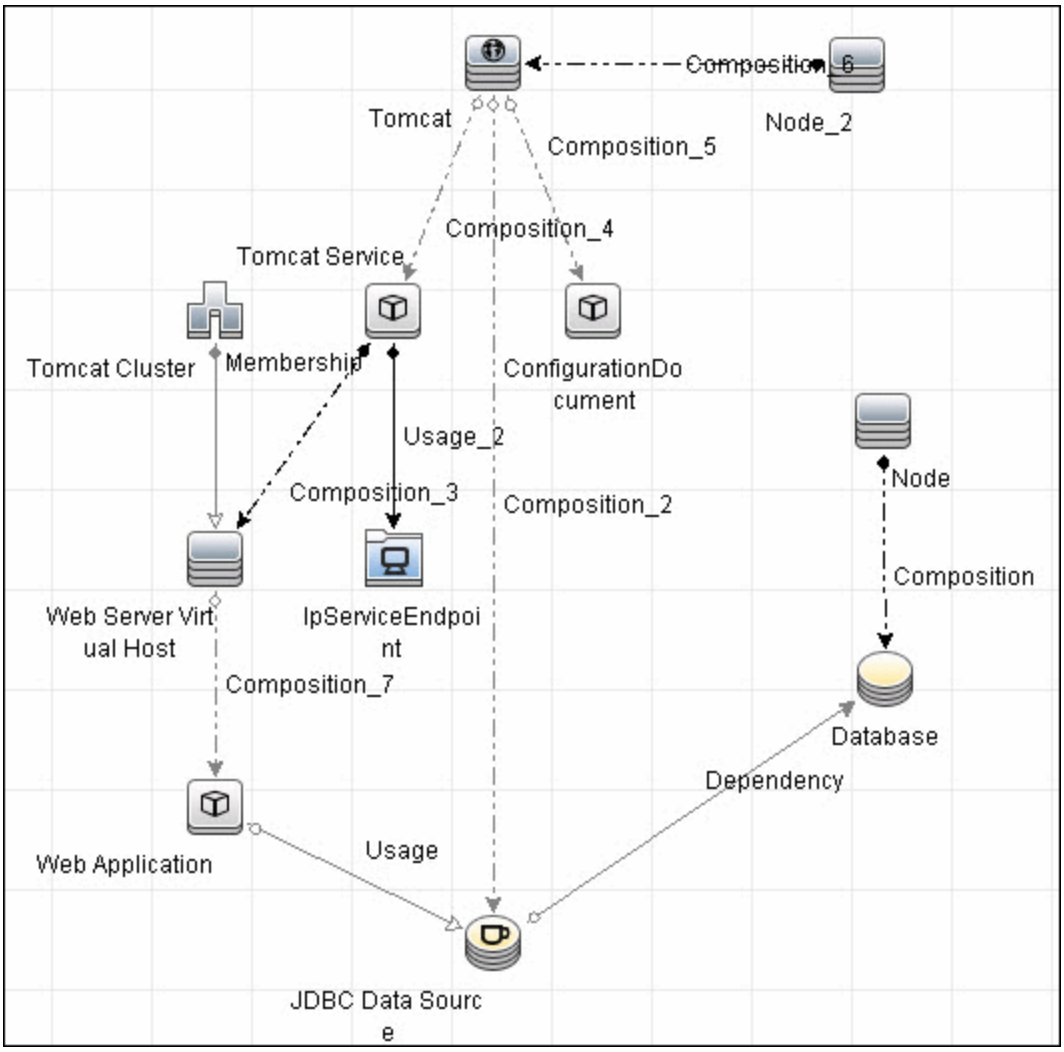
DFM discovers Tomcat running on the following operating systems:

- Windows
- UNIX
- Linux

Topology

The following image displays the topology of the Apache Tomcat discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 1127.



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" on page 82.

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 Resources and 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 "Discovery Control Panel" 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 **WebServer 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)

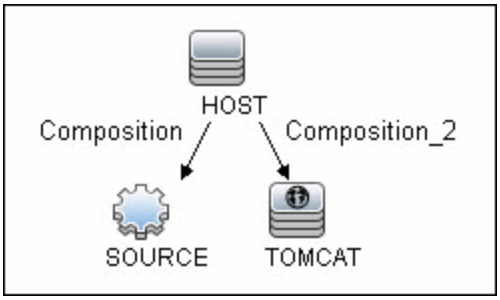
Apache Tomcat by Shell Job

This section includes:

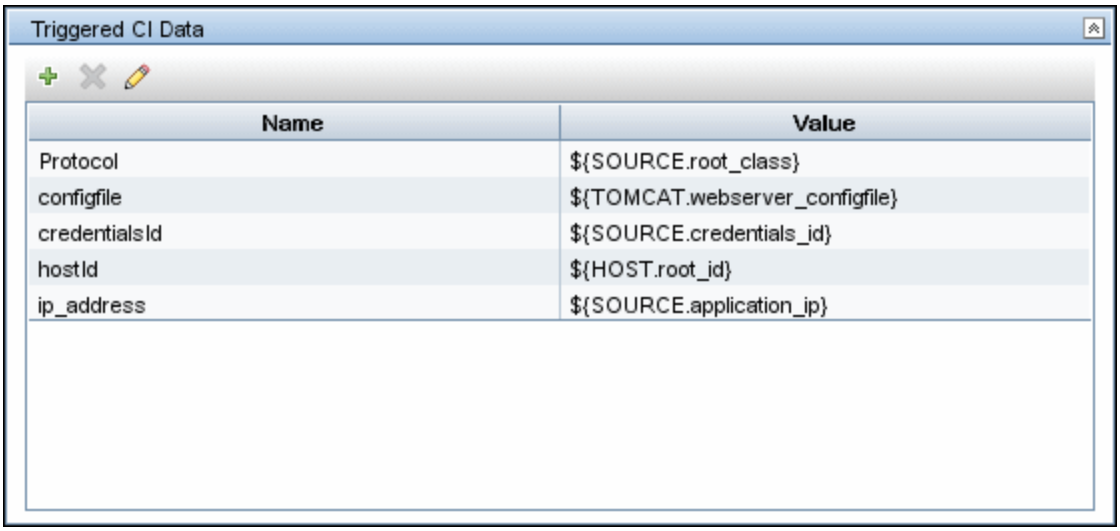
Adapter

This job uses the ApacheTomcat_Topology adapter.

- Input Query



- Triggered CI Data



Discovered CITs

The following CITs are discovered:

- **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 1124.

Chapter 84

Microsoft Internet Information Services (IIS) Discovery

This chapter includes:

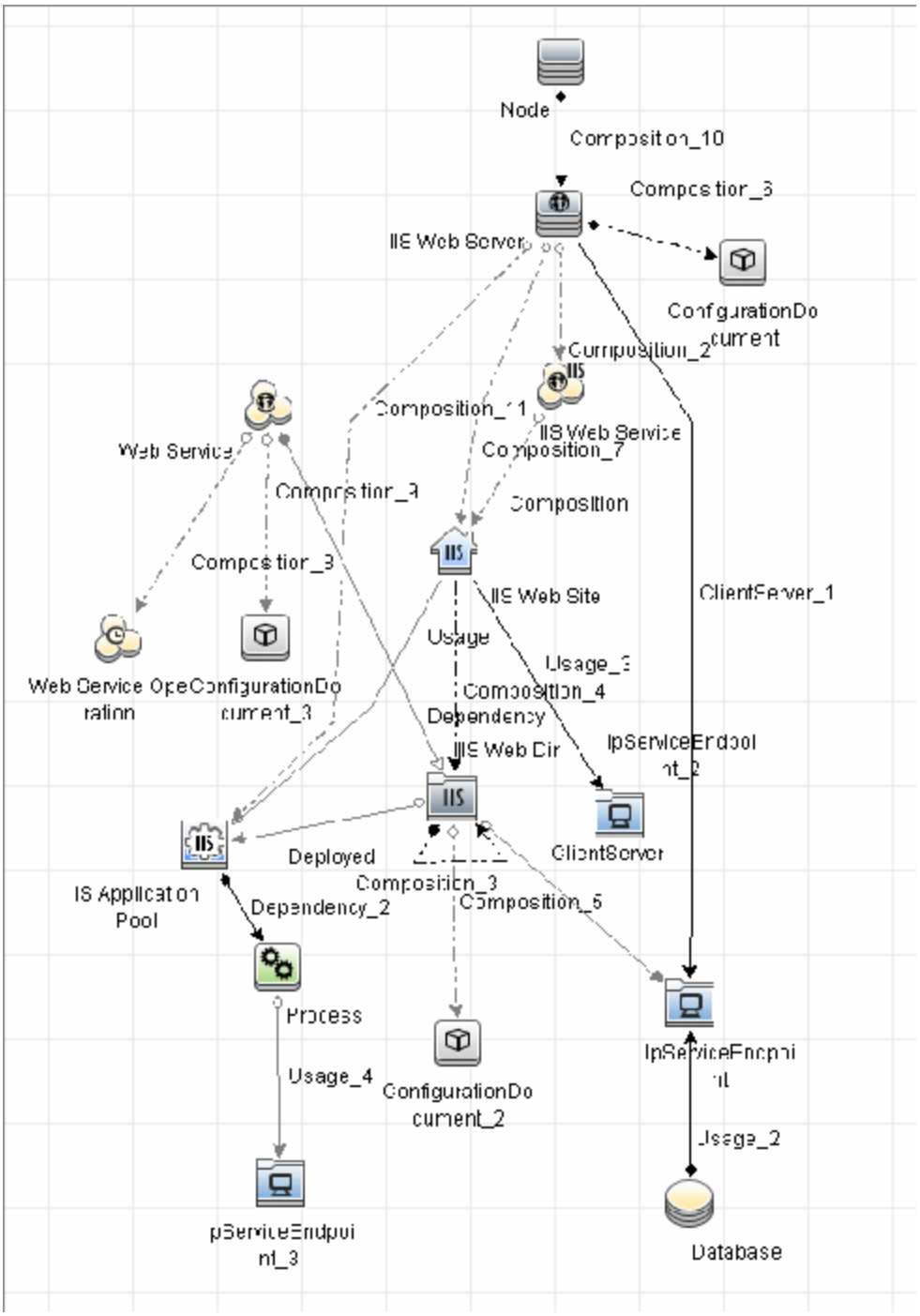
Supported Versions

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 "Discovered CITs" on page 1134.



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" on page 82.

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 Discovery Control Panel window, activate the jobs in the following order:

   a. Run the **Host Connection by Shell** job to create Shell CITs.

   b. Run the **Host Resources and 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: **Discovery Control Panel > Advanced Mode > Web Servers > IIS > IIS Applications by NTCMD or UDA** job. **Details** tab > **Discovery Job Details** pane. Click the **View Permissions** button. For details, see "Permissions" on next page.

      > **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.

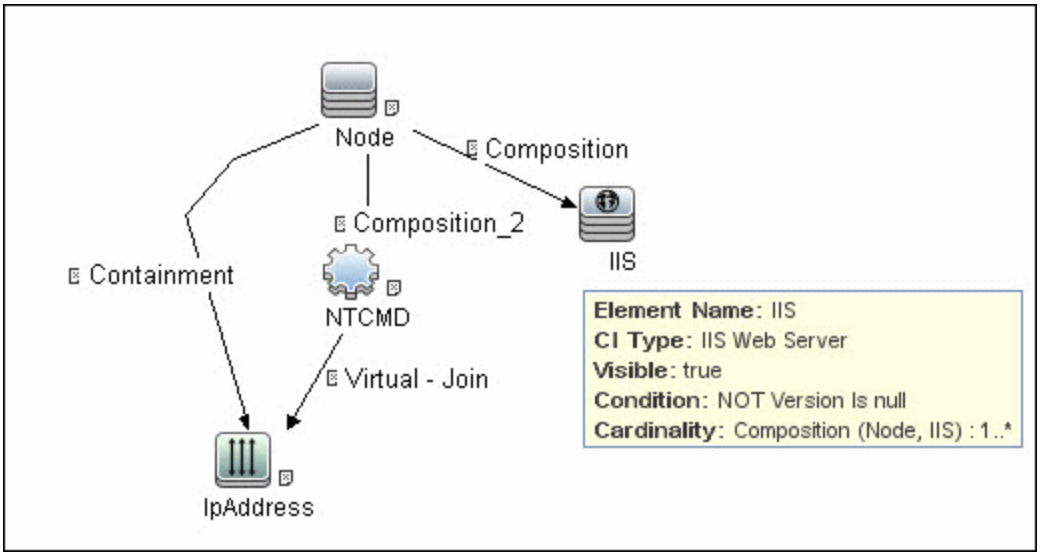
      For details on running jobs, refer to "Discovery Control Panel" in the *HP Universal CMDB Data Flow Management Guide*.

IIS Applications by NTCMD or UDA Job

This section includes:

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

Trigger Query



Adapter

This job uses the NTCMD_APP_Dis_IIS adapter.

- **Triggered CI Data**

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.
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 is 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 1126.

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 XIII: Cloud

Chapter 85

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.

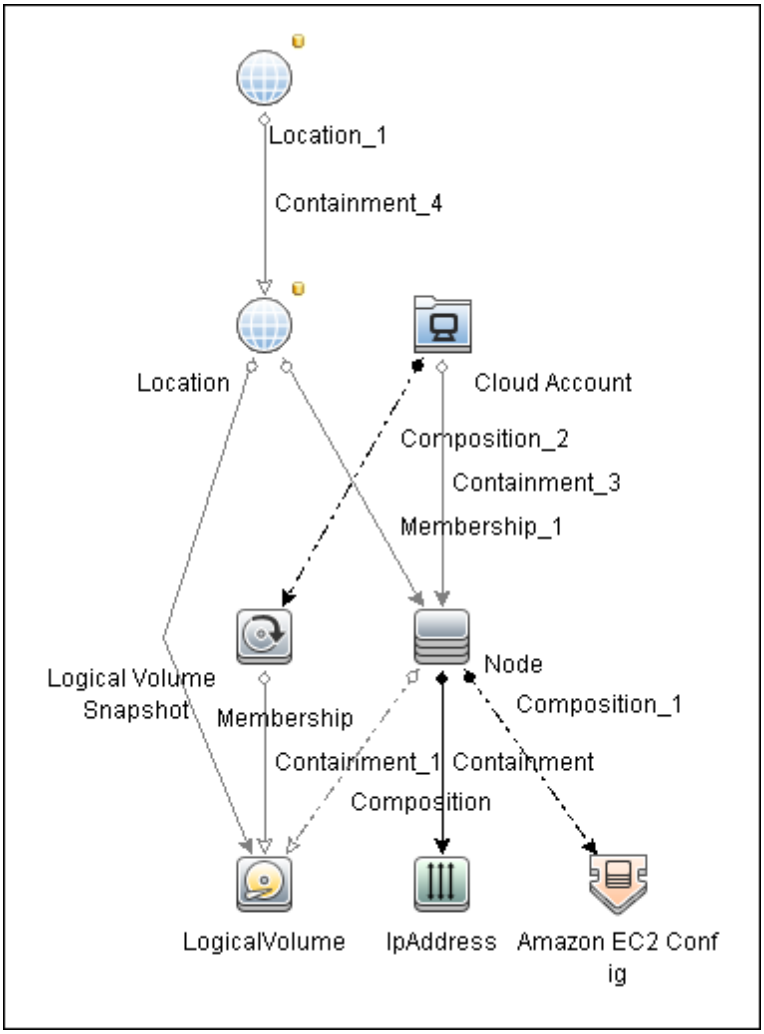
For communication with AWS, discovery uses Amazon SDK and IAM service for the authentication.

Topology

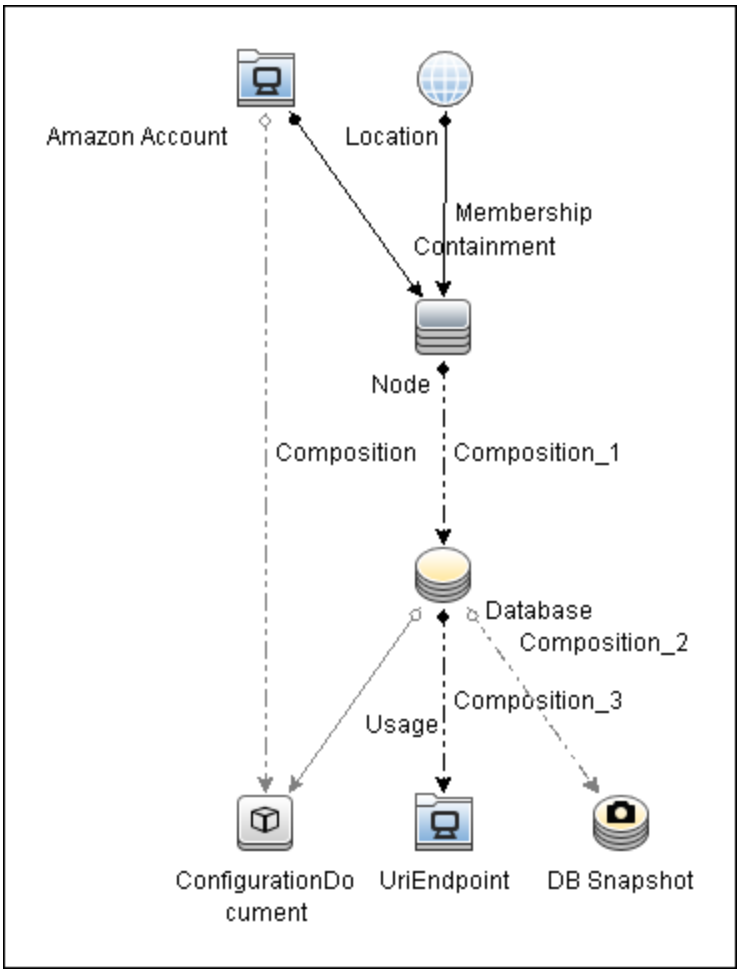
The following images display the topology of AWS discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 1144

Amazon EC2



Amazon RDS


Amazon Account
Location
Membership
Containment
Node
Composition
Composition_1
Database
Composition_2
Composition_3
Usage
ConfigurationDo
cument
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:

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/httpcomponents-client*/*.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

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**
- **jdbc_url_parser.py**
- **jdbc.py**

Discovered CITs

- **Amazon Account**
- **Amazon EC2 Config**
- **Composition**
- **ConfigurationDocument**
- **Containment**

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

AWS by Web Services Job

This section includes:

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 86

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.

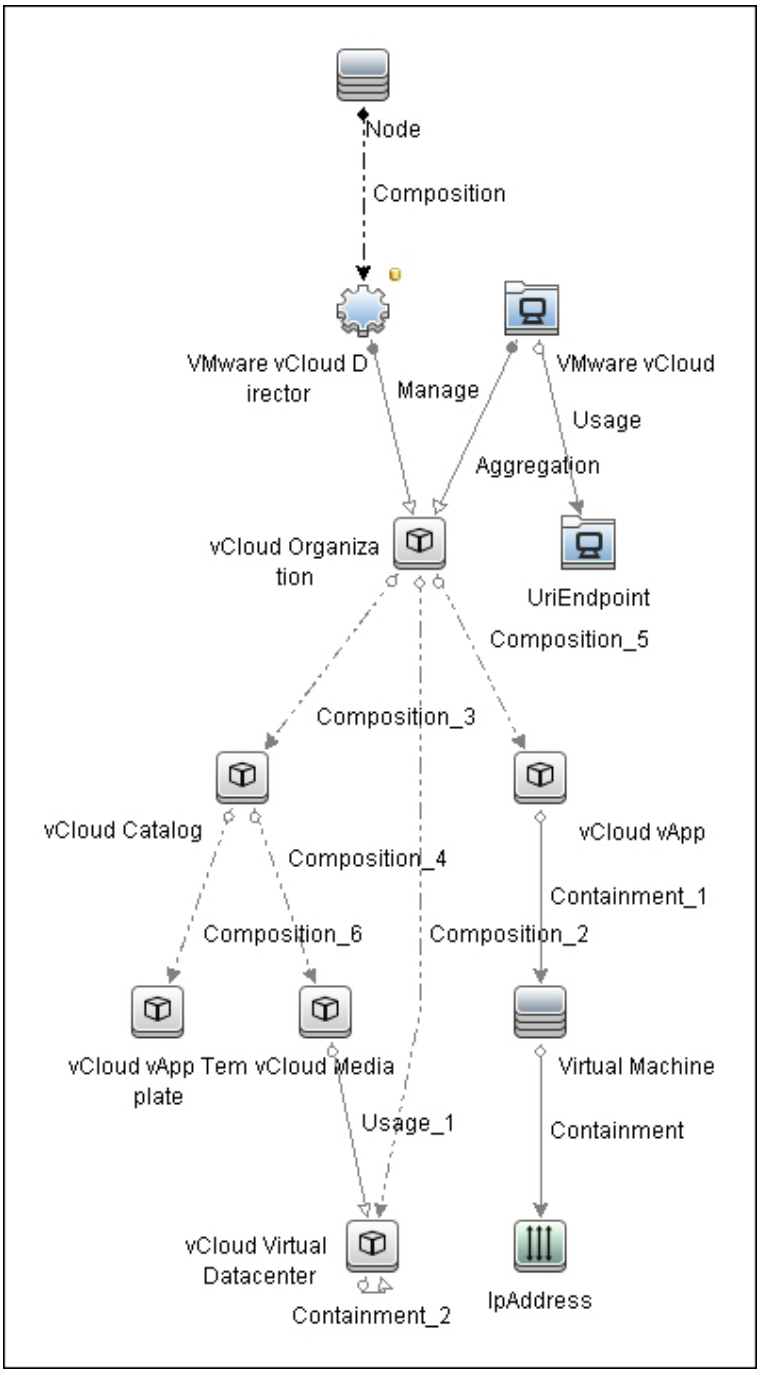
Supported Versions

VMware vCloud Director Version 1.5.

Topology

The following image displays the topology of vCloud discovery.

Note: For a list of discovered CITs, see "Discovered CITs" on page 1154



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 page 1153.
   c. Define the following credentials:
      i. **SSH** or **Telnet**
      ii. **vCloud**

      For credential information, see "Supported Protocols" on page 82.
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 Resources and 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" on next page.
   b. Define the **vCloud** credential.

      For credential information, see "Supported Protocols" on page 82.
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

This task contains the following steps.

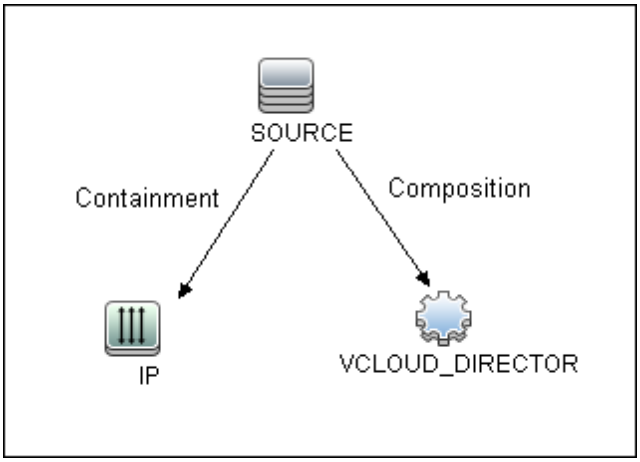
1. Download VMware vCloud SDK 1.5 archive from the VMware community site.
2. Copy the following jar files to the **%PROBE_ROOT%\content\lib** folder:
   - **SDK-1.5.0\rest-api-schemas-1.5.0.jar**
   - **SDK-1.5.0\vcloud-java-sdk-1.5.0.jar**
   - **SDK-1.5.0\libs\httpclient-4.1.jar**
   - **SDK-1.5.0\libs\httpcore-4.1.jar**
3. Restart the probe.

vCloud_Director_by_vCloud_API 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_address	${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 Managed Organization**
- **vCloud Media**
- **vCloud System Organization**
- **vCloud vApp**
- **vCloud vApp Template**
- **vCloud Virtual Datacenter**
- **VMware vCloud**
- **VMware vCloud Director**

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

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 Managed Organization**
- **vCloud Media**
- **vCloud System Organization**
- **vCloud vApp**
- **vCloud vApp Template**
- **vCloud Virtual Datacenter**
- **VMware vCloud**
- **VMware vCloud Director**

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

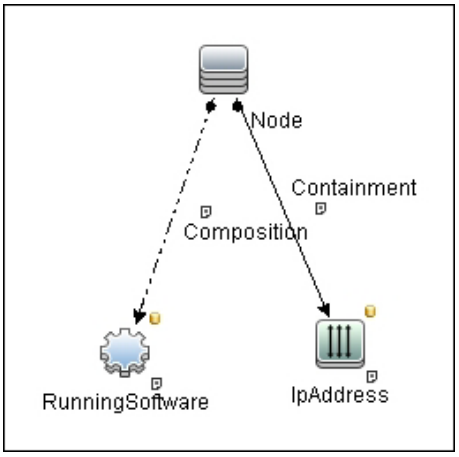
vCloud Director by vCloud API Job

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

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.