HP Business Availability Center

for the Windows and Solaris operating systems

Software Version: 8.03

Discovery and Dependency Mapping

Document Release Date: September 2009
Software Release Date: September 2009

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

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 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

Table of Contents

PART I: THE DISCOVERY AND DEPENDENCY MAPPING PROBE

PART II: INTRODUCTION

PART III: ADMINISTRATION

PART IV: CONTENT WRITING

PART V: DISCOVERY AND DEPENDENCY MAPPING SECURITY

Welcome to This Guide

This guide describes how to install the Discovery and Dependency Mapping (DDM) Probe, how to manage the DDM process and to automatically discover and map IT infrastructure resources and their interdependencies. DDM can discover such resources as applications, databases, network devices, different types of servers, and so on. For users with an advanced knowledge of DDM, there is a section on content writing.

For details on working with DDM content, see *Discovery and Dependency Mapping Content Guide*.

This chapter includes:

- How This Guide Is Organized on page 11
- Who Should Read This Guide on page 12
- Getting More Information on page 13

How This Guide Is Organized

The guide contains the following chapters:

Part I The Discovery and Dependency Mapping Probe

Explains how to install the DDM Probe and provides details on how the Probe works.

Part II Introduction

Describes the main concepts, tasks, and reference for the Run Discovery, Set Up Discovery Probes, Manage Discovery Resources, and Show Status Snapshot applications.

Part III Administration

Explains how to use the DDM applications and the HP Discovery and Dependency Mapping Web Service API (intended for users with an advanced knowledge of Discovery and Dependency Mapping) to manage discovery and dependency.

Part IV Content Writing

Describes the approaches, methodologies, and practices of developing new Discovery and Dependency Mapping (DDM) content (also known as pattern-writing). Also, explains the multi-lingual locale feature.

Part V Discovery and Dependency Mapping Security

This section explains how to harden Discovery and Dependency Mapping and the DDM Probe.

Who Should Read This Guide

This guide is intended for the following users of HP Business Availability Center:

- HP Business Availability Center administrators
- HP Business Availability Center platform administrators
- HP Business Availability Center application administrators
- HP Business Availability Center data collector administrators

Readers of this guide should be knowledgeable about enterprise system administration, have familiarity with ITIL concepts, and be knowledgeable about HP Business Availability Center in general and HP Universal CMDB technology specifically.

Getting More Information

For a complete list of all online documentation included with HP Business Availability Center, additional online resources, information on acquiring documentation updates, and typographical conventions used in this guide, see the *HP Business Availability Center Deployment Guide* PDF.

Part I

The Discovery and Dependency Mapping Probe

1

DDM Probe Installation

This chapter describes the procedures that are needed for the installation of the Discovery and Dependency Mapping (DDM) Probe on a Windows platform.

You cannot install the Probe on a Solaris machine. If the HP Universal CMDB server is installed on a Solaris machine, install the DDM Probes from the Windows CD-ROM.

This chapter includes:

Tasks

Reference

Install the DDM Probe

The following procedure explains how to install the DDM Probe.

The Probe can be installed before or after you install the HP Business Availability Center Gateway server. However, during Probe installation you must provide the server name, so it is preferable to install the server before installing the Probe.

It is recommended to install the Probe on a separate server from the Business Availability Center servers, to distribute the overall system load.

To install the DDM Probe:

1 Select **Admin** > **Platform** > **Setup and Maintenance** > **Downloads**.

Note: The Probe link in the Downloads page is displayed only if you have purchased a standard or advanced license for Discovery and Dependency Mapping, and if the administrator has added the Probe link to the Downloads page. For details, see “Licensing Models for HP Universal CMDB” and “Installing Component Setup Files” in the *HP Business Availability Center Deployment Guide* PDF.

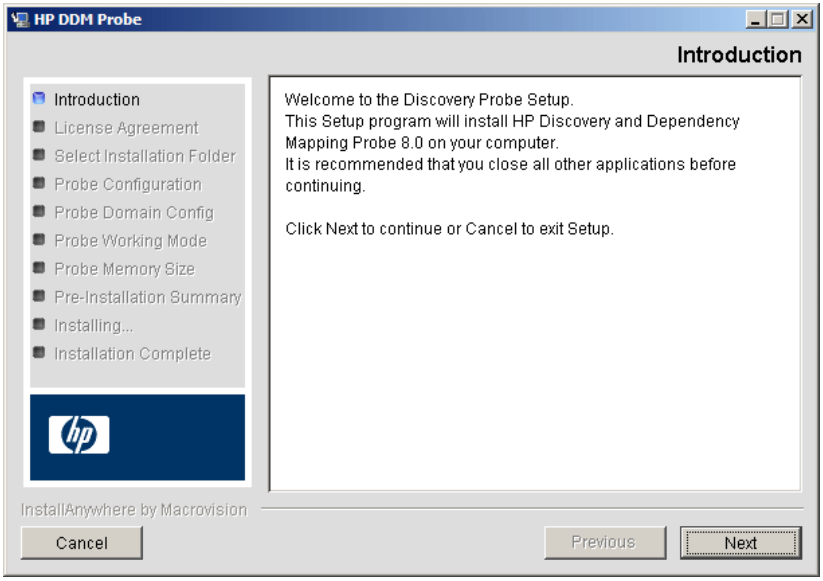
---

2 Click **Discovery Probe for Windows 2000/2003/XP**. You can open the Setup file or save it to your computer:

- ➤ If you choose to open the file, it is not saved to your computer, and the setup program starts immediately. In this case, depending on your browser security settings, a security warning dialog box may open. Confirm that you want to proceed.
- ➤ If you choose to save the file to your computer, double-click the downloaded file to begin installation.

A progress bar is displayed. Once the initial process is complete, the splash screen opens.

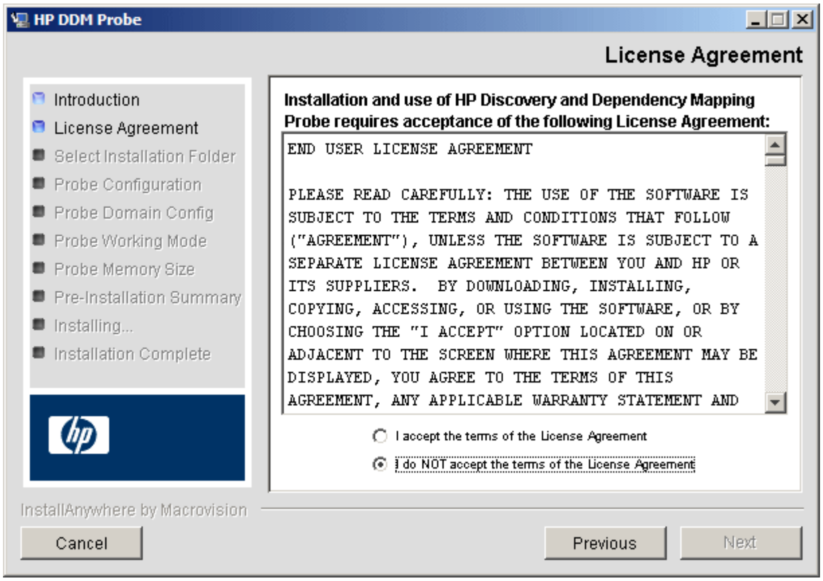
3 Choose the locale language and click **OK**. The Introduction dialog box opens.



Welcome to the Discovery Probe Setup.
This Setup program will install HP Discovery and Dependency Mapping Probe 8.0 on your computer.
It is recommended that you close all other applications before continuing.

Click Next to continue or Cancel to exit Setup.

4 Click **Next** to open the License Agreement dialog box.



Installation and use of HP Discovery and Dependency Mapping Probe requires acceptance of the following License Agreement:

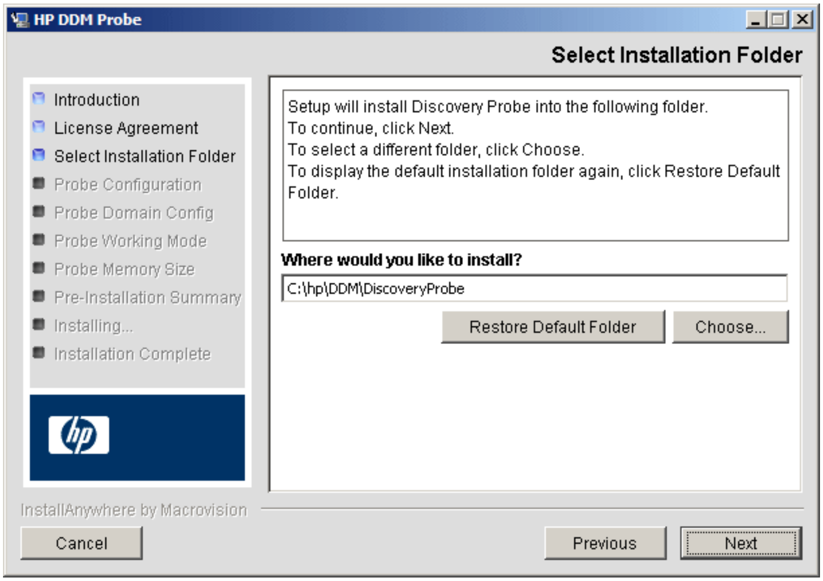
```
END USER LICENSE AGREEMENT

PLEASE READ CAREFULLY: THE USE OF THE SOFTWARE IS
SUBJECT TO THE TERMS AND CONDITIONS THAT FOLLOW
("AGREEMENT"), UNLESS THE SOFTWARE IS SUBJECT TO A
SEPARATE LICENSE AGREEMENT BETWEEN YOU AND HP OR
ITS SUPPLIERS.  BY DOWNLOADING, INSTALLING,
COPYING, ACCESSING, OR USING THE SOFTWARE, OR BY
CHOOSING THE "I ACCEPT" OPTION LOCATED ON OR
ADJACENT TO THE SCREEN WHERE THIS AGREEMENT MAY BE
DISPLAYED, YOU AGREE TO THE TERMS OF THIS
AGREEMENT, ANY APPLICABLE WARRANTY STATEMENT AND
```

○ I accept the terms of the License Agreement

● I do NOT accept the terms of the License Agreement

5 Accept the terms of the agreement and click **Next** to open the Select Installation Folder dialog box.



Setup will install Discovery Probe into the following folder.
To continue, click Next.
To select a different folder, click Choose.
To display the default installation folder again, click Restore Default Folder.

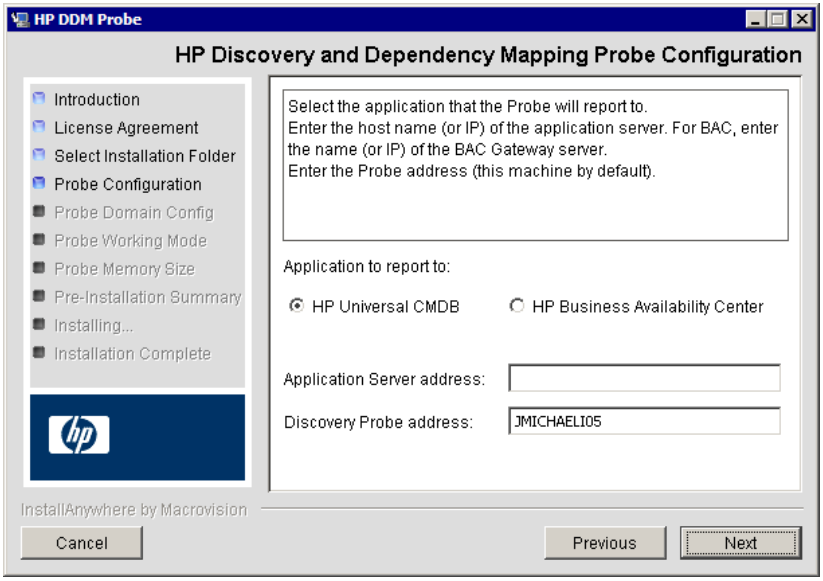
Where would you like to install?

C:\hp\DDM\DiscoveryProbe

Accept the default entry or click **Choose** to display a standard Browse dialog box. To install to a different directory, browse to and select the installation folder.

Note: To restore the default installation directory, after selecting a directory in the Browse dialog box, click **Restore Default Folder**.

6 Click **Next** to open the HP Discovery and Dependency Mapping Probe Configuration dialog box.



HP DDM Probe

HP Discovery and Dependency Mapping Probe Configuration

Select the application that the Probe will report to.
Enter the host name (or IP) of the application server. For BAC, enter the name (or IP) of the BAC Gateway server.
Enter the Probe address (this machine by default).

Application to report to:

● HP Universal CMDB ○ HP Business Availability Center

Application Server address:

Discovery Probe address: JMICHAELI05

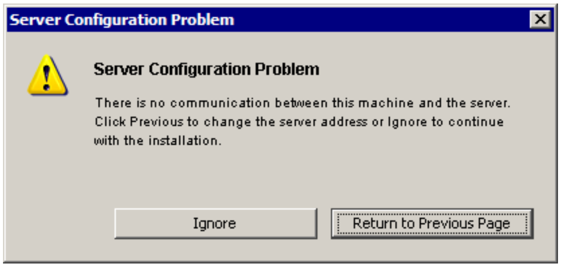
- **Application to report to.** Choose the application server with which you are working. You can use the Probe with either HP Universal CMDB or HP Business Availability Center.

  - If you select HP Universal CMDB, in the **Application Server address** box, enter the name or the IP address of the HP Universal CMDB server to which the Probe is to be connected.

  - If you select HP Business Availability Center, in the **Application Server address** box, enter the IP or the DNS name of the Gateway Server.

- In the **Discovery Probe address** box, enter the IP address or the DNS name of the machine on which you are currently installing the Probe, or accept the default.

7 If you do not enter the address of the application server, a message is displayed. You can choose to continue to install the Probe without entering the address, or to return to the previous page and add the address.

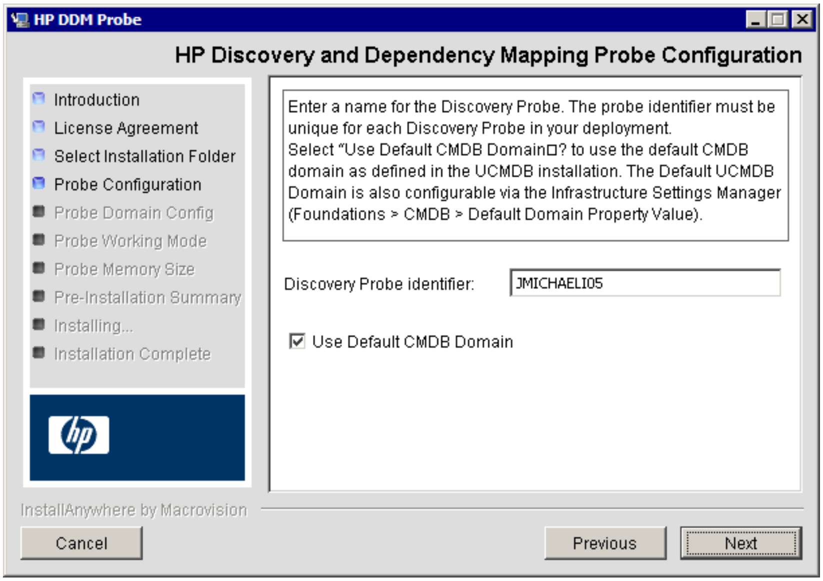


Server Configuration Problem

Server Configuration Problem

There is no communication between this machine and the server. Click Previous to change the server address or Ignore to continue with the installation.

8 Click **Next** to open the HP Discovery and Dependency Mapping Probe Configuration dialog box.



HP DDM Probe

HP Discovery and Dependency Mapping Probe Configuration

Enter a name for the Discovery Probe. The probe identifier must be unique for each Discovery Probe in your deployment.
Select "Use Default CMDB Domain□? to use the default CMDB domain as defined in the UCMDB installation. The Default UCMDB Domain is also configurable via the Infrastructure Settings Manager (Foundations > CMDB > Default Domain Property Value).

Discovery Probe identifier: JMICHAELI05

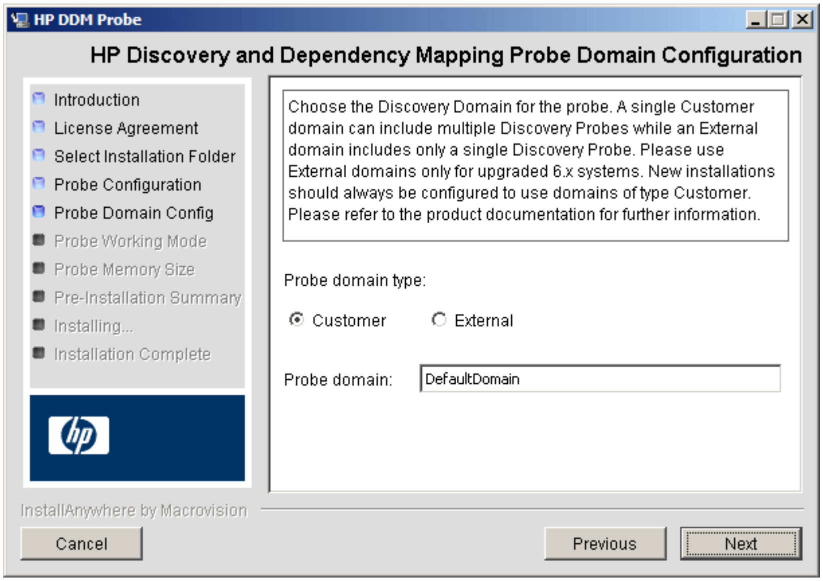
☑ Use Default CMDB Domain

➤ In the **Discovery Probe identifier** box, enter a name for the Probe that will be used to identify it in your environment.

Important: The UCMDB Probe identifier must be unique for each Probe in your deployment.

- Select **Use Default CMDB Domain** to use the default UCMDB IP address or machine name, as defined in the BAC Server installation. The Default UCMDB Domain is also configurable via the Infrastructure Settings Manager, available after installing HP Universal CMDB (**Foundations** > **CMDB** > **CMDB Class Model Settings** > **Default Domain Property Value**).

9 Click **Next** to open the HP Discovery and Dependency Mapping Probe Domain Configuration dialog box. (This dialog box is displayed only if you cleared the **Use Default CMDB Domain** box in the HP Discovery and Dependency Mapping Probe Configuration dialog box.)



HP Discovery and Dependency Mapping Probe Domain Configuration

Choose the Discovery Domain for the probe. A single Customer domain can include multiple Discovery Probes while an External domain includes only a single Discovery Probe. Please use External domains only for upgraded 6.x systems. New installations should always be configured to use domains of type Customer. Please refer to the product documentation for further information.

Probe domain type:

● Customer ○ External

Probe domain: DefaultDomain

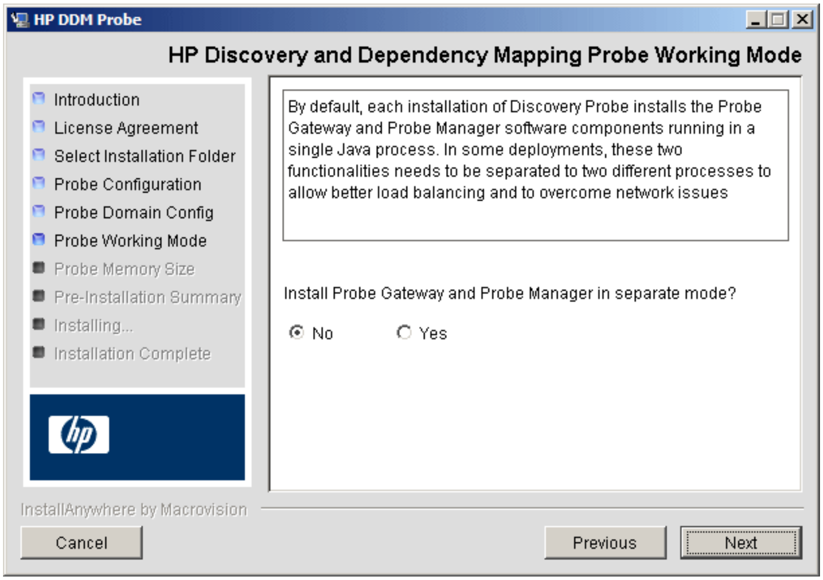
- Choose between **Customer** and **External**, depending on the type of domain on which the Probe is to be running:
  - **Customer.** Select if you are installing one or more Probes in your deployment.

Important: For new installations, always select **Customer**.

  - **External.** Select if you are upgrading from version 6.x systems.

- **Probe domain**: If you are not installing the Probe on the UCMDB Server domain, enter the name of the domain here.

10 Click **Next** to open the HP Discovery and Dependency Mapping Probe Working Mode dialog box.



HP Discovery and Dependency Mapping Probe Working Mode

By default, each installation of Discovery Probe installs the Probe Gateway and Probe Manager software components running in a single Java process. In some deployments, these two functionalities needs to be separated to two different processes to allow better load balancing and to overcome network issues

Install Probe Gateway and Probe Manager in separate mode?

● No ○ Yes

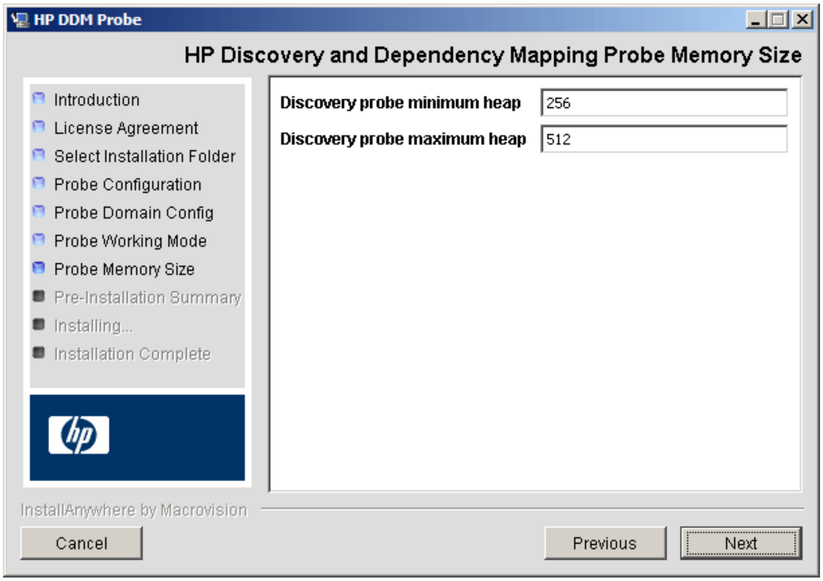
You can run the Probe Gateway and Manager as one Java process or as separate processes. You would probably run them as separate processes in deployments that need better load balancing and to overcome network issues.

Click **No** to run Probe Gateway and Probe Manager as one process.

Click **Yes** to run Probe Gateway and Probe Manager as two processes. For details on the procedure, see “Run Probe Manager and Probe Gateway on Separate Machines” on page 27.

For details on Probe Gateway and Probe Manager, see “DDM Probe Tasks” on page 31.

11 Click **Next** to open the HP Discovery and Dependency Mapping Probe Memory Size dialog box.



HP DDM Probe

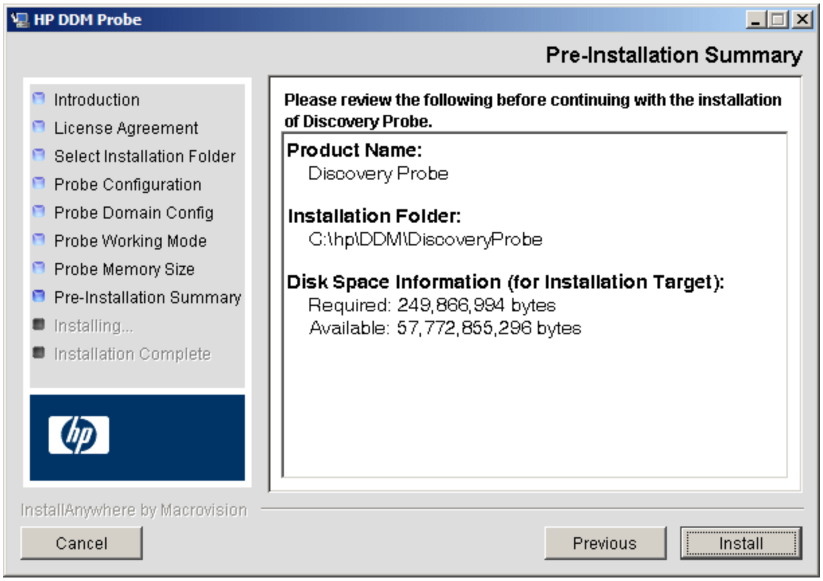
HP Discovery and Dependency Mapping Probe Memory Size

Discovery probe minimum heap: 256

Discovery probe maximum heap: 512

Define the minimum and maximum memory to be allocated to the Probe. The values are measured in megabytes.

12 Click **Next** to open the Pre-Installation Summary dialog box and review the selections you have made.



Please review the following before continuing with the installation of Discovery Probe.

Product Name:
Discovery Probe

Installation Folder:
C:\hp\DDM\DiscoveryProbe

Disk Space Information (for Installation Target):
Required: 249,866,994 bytes
Available: 57,772,855,296 bytes

13 Click **Install** to complete the installation of the Probe. When the installation is complete the Install Complete page is displayed.

Note: Any errors occurring during installation are written to the following file: **C:\hp\DDM\DiscoveryProbe\Discovery_Probe_InstallLog.log**.

14 Click **Done**. The following shortcut is added to the Windows **Start** menu:

Programs > **HP DDM** > **DDM Probe**

15 Activate the Probe by selecting the shortcut.

The Probe is displayed in HP Universal CMDB: access **Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes** > **Domains and Probes**. For details, see “Domains and Probes Pane” on page 200.

Upgrade the Probe

This task describes how to upgrade the DDM Probe.

1 Uninstall the Old Probe

Uninstall all existing Probes. If a Probe is running, stop it before you uninstall it.

2 Install the New Probe

For details on installation, see "Install the DDM Probe" on page 18.

Note:

➤ You should install the new Probe with the same configuration, that is, use the same Probe ID, domain name, and server name as for the previous Probe installation.

➤ You must reactivate active jobs after the upgrade so that the newly installed Probes receive the tasks they are assigned.

Run Probe Manager and Probe Gateway on Separate Machines

During installation, you can choose to separate the Probe Manager and Probe Gateway processes so that they run on separate machines. You must:

1. Install the Probe on both machines according to the procedure in "Install the DDM Probe" on page 18.
2. Choose **Yes** in step 10 on page 24.
3. Perform the configuration in "Configure the Probe Manager and Probe Gateway Components" on page 28.

Configure the Probe Manager and Probe Gateway Components

This section explains how to set up the Probe when the Probe Manager and Probe Gateway run as separate processes on two machines.

This section includes the following topics:

- “Set Up the Probe Gateway Machine” on page 28
- “Set Up the Probe Manager Machine” on page 29
- “Start the Services” on page 29

1 Set Up the Probe Gateway Machine

a Open the following file: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeGateway\probeMgrList.xml**.

b Locate the line beginning <**probeMgr ip**>= and add the Manager machine name in uppercase, for example:

```
<probeMgr ip>=OLYMPICS08
```

c Open the following file: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties**.

d Locate the lines beginning **appilog.collectors.local.ip** = and **appilog.collectors.probe.ip** = and enter the Gateway machine name in upper case, for example:

```
appilog.collectors.local.ip = STARS01
appilog.collectors.probe.ip = STARS01
```

2 Set Up the Probe Manager Machine

a In **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties**, locate the line beginning **appilog.collectors.local.ip** = and enter the Manager machine name in uppercase, for example:

```
appilog.collectors.local.ip = OLYMPICS08
```

b Locate the line beginning **appilog.collectors.probe.ip** = and enter the Gateway machine name in uppercase, for example:

```
appilog.collectors.probe.ip = STARS01
```

3 Start the Services

a On the Probe Manager machine start the Manager: **Start** > **Programs** > **HP DDM** > **DDM Manager.**

b On the Probe Gateway machine start the Gateway: **Start** > **Programs** > **HP DDM** > **DDM Gateway**.

Probe Installation Requirements

Hardware Requirements

Computer/processor	**Windows:** Pentium IV 2.4 GHz or later processor
Memory	**Windows:** Minimum 1 GB RAM (Recommended: 2 GB RAM)
Virtual memory (for Windows deployment)	Minimum 2 GB **Note:** The virtual memory size should always be at least twice the physical memory size.
Free hard disk space	**Windows:** Minimum 4 GB (at least 4 GB for database software and data files) (Recommended: 20 GB hard disk)
Display	**Windows:** Color palette setting of at least 256 colors (32,000 colors recommended)
Installation on a virtual machine	The installation of the DDM Probe on a virtual machine is supported for UCMDB version 8.00 or later

Software Requirements

Operating system	**Windows:** ➤ Windows 2000 Server/Advanced Server, Service Pack 4 or later ➤ Windows 2003 Standard/Enterprise editions, Service Pack 1, Service Pack 2
Java Runtime Environment	JRE 1.5.0 (installed with the product)

2

Introduction to the DDM Probe

This chapter provides information on the DDM Probe.

This chapter includes:

Concepts

Tasks

Reference

DDM Probe Tasks

This section explains how the DDM Probe manages tasks.

The DDM Probe comprises two components: the Probe Gateway and the Probe Manager.

- The Probe Gateway communicates with the server via HTTP or HTTPS, for processes such as downloading tasks and returning task results. The connection is always issued by the Probe Gateway.
- The Probe Manager runs the DDM process itself.

The Probe Gateway communicates with the Probe Manager using RMI.

By default, the Gateway and Manager run as a single process but they can be configured (during installation) to reside on separate processes. For details, see step 10 on page 24 in "Install the DDM Probe." Moreover, several Probe Managers can be configured to connect to a single Probe Gateway. This can be useful if a specific Probe Manager must deal with certain DDM jobs.

This section includes the following topics:

- "Stage 1. Probe Gateway" on page 32
- "Stage 2. Business Availability Center Server" on page 32
- "Stage 3. Probe Gateway" on page 33
- "Stage 4. Probe Manager" on page 33
- "Stage 5. Probe Gateway" on page 34
- "Stage 6. Probe and Server Synchronization Process" on page 35
- "Probe Configuration Update" on page 35

Stage 1. Probe Gateway

The Business Availability Center server does not initiate tasks on the Probe; it is the Probe's responsibility to request relevant tasks to run.

Stage 2. Business Availability Center Server

At the same time as the Probe requests tasks from the Gateway Server, it also sends to the server the last update time of its configuration and the last task ID received. The server returns to the Probe one of the following:

- Updated server data (in the case that the configuration on the Probe is not current). The server data includes: Python scripts, patterns, the Domain Scope Document dictionary file, and so on. For details, see "Probe Configuration Update" on page 35. For details on using the Domain Scope Document to harden DDM, see Chapter 13, "Hardening DDM."
- The last task sent (if there is a mismatch between the Probe and the server last task ID).

- New tasks to run (if any exist):
  - If a job has been deactivated, the server sends a **delete job** message to the Probe.
  - If a job has been activated, the server sends a **run new job** message to the Probe.
- The Business Availability Center Gateway server sends the Probe a response (in XML format) with the new task data. Each task contains the job and pattern names, and the relevant Trigger CI data.

  The number of Trigger CIs for a task is limited (100 by default). For example, if an active job includes 1000 destinations, the job is sent to the Probe in 10 tasks with 100 Trigger CIs in each task.

Stage 3. Probe Gateway

- When the Probe Gateway receives tasks from the Business Availability Center server, it saves them to its local database (MySQL).
- Periodically, a thread on the Probe Gateway scans the database for tasks and sends them to the Probe Manager. This process enables load balancing in DDM when there are multiple Probe Managers for each Probe Gateway.

Stage 4. Probe Manager

- The tasks on the Probe Manager are scheduled using the Quartz third-party library. When tasks are completed, the Probe Manager sends the results (in XML format) to the Probe Gateway. (For details on Quartz, refer to the documentation at http://www.opensymphony.com/quartz/.)
- The Probe Manager receives a set of result objects. The Probe Manager first performs processing on the results (for example, filters results, runs the result redundant mechanism), and only then prepares the results for sending to the Probe Gateway.
- The results are stored in the Probe Manager database.

- A thread scans the database for results that are ready to be sent to the Probe Gateway. These results are merged into a single result, whose size does not exceed a maximum result size (currently 20,000), as defined in the discoveryProbe.properties file:

  ```
  appilog.agent.local.maxTaskResultSize = 20000
  ```

  When results reach the Gateway, it immediately responds with a success or failure reply. Based on this acknowledgement from the Gateway, the Probe Manager marks the results as **ack** in the database, so that they are not sent again during the next cycle.

- The task results that have been acknowledged by the Gateway remain in the Probe Manager database till they are deleted—once a week.
- When results reach the Probe Gateway, they are not sent directly to the server, but are stored in the Gateway database, to avoid flooding the server with data.

Stage 5. Probe Gateway

- A dedicated thread on the Probe Gateway scans the database and searches for task results that are ready to be sent to the server. These results are sent to the server by the Probe Gateway, using the **sendResultsToServer()** API.
- If the size of data that needs to be sent is too large, it is sent in chunks (max. 50,000). The information is then updated in the CMDB (using create, update, or remove).
- Finally, the Probe Gateway verifies that the server has finished handling the results, deletes those results from its database (so they are not sent again to the server), and continues sending results to the server, if any more results exist.

Stage 6. Probe and Server Synchronization Process

After reading a predefined number of tasks, the Probe confirms these tasks with the server. (This process prevents the need for manually reactivating patterns or jobs.)

- The Probe sends to the server the names of all activated jobs and the number of Trigger CIs for each job.
- The server checks that the number matches that in the CMDB:
  - If a job is missing from the Probe, the server redispatches the job to the Probe.
  - If the Probe has less or more than the number of CIs on the server, the server returns the names of the problematic jobs and their CIs to the Probe.
- The Probe checks the problematic jobs' list. If the Probe includes a job that does not exist on the server, the Probe sends a **remove job** remote method call to all Probe Managers.
- If the Probe includes a CI that does not exist on the server, the Probe sends a **remove CI** remote method call to all Probe Managers.
- If the server includes a Trigger CI that does not exist on the Probe, the Probe requests this CI from the server. The server returns the task (in XML format) and the Probe distributes this task.

Probe Configuration Update

- To perform discovery, the Probe needs resource data, such as the Domain Scope Document dictionary file, scripts, and so on. The Probe is updated automatically with these resources. Along with each task request from the Probe Gateway to the server, the Probe Gateway sends the last (server) update time of its latest updated resources.
- The server, before returning any new tasks, validates that there are no more recently updated resources. If there are, instead of returning the regular queued tasks for the Probe, the server returns a special, crafted task for updating the Probe's resources.

- When the Probe receives this task, it sends a **GetResouces()** request to the server, which returns a list of resources that have not been updated to the Probe. In that way the Probe is always updated with the latest system configuration files.

Data Validation on the DDM Probe

From version 7.0, the CIT model also resides on the DDM Probe. This enables data validation to take place on the Probe when receiving data from services. Problems are generated for a specific Trigger CI and displayed to the user. For details, see "Discovery Status Pane" on page 135.

The following validation takes place on the Probe:

- The CIT of the CI is compared to that in the CIT model.
- The CI is checked to verify that all key attributes are present (on condition that the CmdbObjectId attribute is not defined).
- The CI's attributes are checked to verify that they are all defined in the CIT.
- The CI's attributes of type STRING are checked to verify that they do not exceed the size limit. If an attribute is longer than the limit, DDM checks whether an AUTO_TRUNCATE qualifier is defined for the attribute. If there is a qualifier, the value is truncated and a warning message is written to the Probe error.log file.

All invalid attributes raise a CollectorsProcessException exception, which reports on a specific CI. When the Probe finds invalid data that is related to the CITs, all data that the Probe has collected on that CI is dropped by the Probe and is not sent to the server.

For details on attributes, see "CI Type Attributes" in *Model Management*.

Hardening the DDM Probe

For details on hardening the DDM Probe, see Chapter 14, “Hardening the DDM Probe.”

For details on hardening the Domain Scope Document, see “Control the Location of the domainScopeDocument File” on page 402.

For details on enabling SSL on the DDM Probe, see “Using SSL with the DDM Probe” on page 403.

Filtering Results

You can filter results sent by the Probe to the Business Availability Center Gateway server. You would probably need to filter irrelevant data regularly during production runs and specifically when you are testing a limited environment.

There are two levels of filtering: pattern filtering and global filtering:

➤ **Pattern filtering.** DDM filters the results for a specific pattern and sends to the CMDB only those filtered CIs. You define a pattern filter in the Results Management Pane in the Pattern Management tab. For details, see “Pattern Management Tab” on page 258.

➤ **Global filtering.** DDM filters the results of all jobs running on a Probe. You define global filters in the globalFiltering.xml file. For details, see “globalFiltering.xml” on page 236.

The order of filtering is as follows: during a run, DDM first searches for a pattern filter and applies the filter to the results of the run. If there are no pattern filters, DDM searches for a global filter and applies that filter to the results. If DDM finds no filters, all results are sent to the server.

Get Started With the DDM Probe

This section describes how to install and launch the DDM Probe.

Note:

- The Probe link in the Downloads page is displayed only if you have purchased a license for the DDM application.
- The managed environment is defined by the IP ranges of the domains. However, with some patterns it is possible to override this behavior and discover CIs that are out of a Probe's range.

This task includes the following steps:

- "Install the Probe" on page 38
- "Launch the Probe" on page 38
- "Run Discovery and Dependency Mapping" on page 39

1 Install the Probe

For details, see Chapter 1, "DDM Probe Installation."

2 Launch the Probe

On the machine on which the Probe is installed, select **Start** > **Programs** > **HP DDM** > **DDM Probe** to start the Probe. A Command Prompt window opens. To verify that the Probe has been launched successfully, in Business Availability Center select **Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes**. Select the Probe and, in the Details pane, verify that the status is **connected**.

For details on how the Probe works, see "DDM Probe Tasks" on page 31.

Note: To stop the Probe, in the Command Prompt window, press CTRL+C, then **y**.

3 Run Discovery and Dependency Mapping

For details, see “Run Discovery – Overview” on page 94.

The DiscoveryProbe.properties File

A DDM process needs several parameters to be activated. These parameters specify the method to be used (for example, ping five times before declaring a failure) and on which CI a method should be used. If parameters have not been defined by the user, the DDM process uses the default parameters defined in the **DiscoveryProbe.properties** file. To edit the parameters, open **DiscoveryProbe.properties** in a text editor.

The DiscoveryProbe.properties file is located in **C:\hp\DDM\DiscoveryProbe\root\lib\collectors**.

Note: If you update the **DiscoveryProbe.properties** parameters, you must restart the Probe so that it is updated with the changes.

The **DiscoveryProbe.properties** file is divided into the following sections:

- **Server Connection Definitions.** Contains parameters that are needed to set up the connection between the server and the Probe, such as the protocol to be used, machine names, default Probe and domain names, time-outs, and basic authentication.
- **Discovery Probe Definitions.** Contains parameters that define the Probe, such as root folder location, ports, and Manager and Gateway addresses.

- **Probe Gateway Configurations.** Contains parameters that define time intervals for retrieving data.
- **Probe Manager Configurations.** Contains parameters that define Probe Manager functionality, such as scheduled intervals, result grouping, chunking, threading, time-outs, and filtering.
- **I18N Parameters.** Contains parameters that define language settings.
- **Internal Configurations.** (Caution: These parameters should not be changed without an advanced knowledge of Discovery and Dependency Mapping.) Contains parameters that enable DDM to function efficiently, such as thread pool size.

3

Licensing Models for HP Universal CMDB

This chapter provides information on the HP Universal CMDB licensing models.

This chapter includes:

Concepts

Tasks

Licensing Models – Overview

There are three levels of licensing. Each level includes specific packages, as follows:

License Level	Packages
Foundation	➤ NNM_Integration ➤ SE_Integration ➤ Oracle HP Universal CMDB can work with the following packages, but they are not provided by default. For details, contact HP Software Support: ➤ DDMI_Integration ➤ SAR_Integration ➤ SM_Integration
Standard	➤ NNM_Integration ➤ SE_Integration ➤ Oracle ➤ Credential_Less_Discovery ➤ Host_Resources_By_NTCMD ➤ Host_Resources_By_SNMP ➤ Host_Resources_By_TTY ➤ Host_Resources_By_WMI ➤ Layer2 ➤ Network ➤ UMDB7-SCSM HP Universal CMDB can work with the following packages, but they are not provided by default. For details, contact HP Software Support: ➤ DDMI_Integration ➤ SAR_Integration ➤ SM_Integration
Advanced	All packages

This section includes the following topics:

- "License for HP Software-as-a-Service" on page 43
- "License for HP ServiceCenter/Service Manager and Other Integrations" on page 43
- "The DDM Probe Installation" on page 44

License for HP Software-as-a-Service

All packages.

License for HP ServiceCenter/Service Manager and Other Integrations

The following packages are available:

- UCMDB7-SCSM
- SAR_Integration
- DDMI_Integration
- NNM_Integration
- SE_Integration
- SM_Integration

Note:

- The name of the license file is ucmdb_license.xml.
- You are asked for the location of the license file during installation. The default location of the foundation license is: **<Business Availability Center root directory>\mam_lib\server** on the Data Processing server machine.
- Two weeks before the license expires, a reminder message is displayed for you to renew the license.

The DDM Probe Installation

You should install the DDM Probe, no matter which license you are running. If you have the Foundation license, the Probe is needed to run the Integration jobs (NNMi, SE, and DDMi). For details, see "DDM Probe Installation" on page 17.

Upgrade to the Standard or Advanced License

When you install Business Availability Center, you receive the Universal CMDB Foundation license. To obtain the file needed to upgrade to the Standard or Advanced license, contact HP Software Support, then perform the following procedure:

To upgrade your license:

1. Obtain the appropriate file (standard or advanced) from HP Software Support.
2. Replace the **ucmdb_license.xml** file in the **<Business Availability Center root directory>\mam_lib\server** folder on the Data Processing server machine.

   If Business Availability Center is installed in a distributed deployment, replace the file on the Gateway Server machine.
3. Use the JMX console to force a license change:

   a. Launch a Web browser and enter the address **http://<server_name>.<domain_name>:8080/jmx-console**, where **<server_name>** is the name of the machine on which BAC is installed. When prompted, enter the JMX Console authentication credentials (if you do not have these credentials, contact your system administrator).

   b. Under MAM, click **service=UCMDB UI** to open the JMX MBEAN View page.

c Locate **java.lang.String getLicenseForCustomer()** and enter the following information:

In the force parameter box, select **True**.

In the ParamValue box for the parameter **customerId**, enter **1**.

Click **Invoke**.

Note: To verify the type of license that is installed, select **False** and enter the customer ID. Details about the license are displayed.

Troubleshooting and Limitations

This section describes troubleshooting and limitations for UCMDB licensing.

- **Problem**: When integrating UCMDB with HP Storage Essentials, unable to run the **SE Integration by SQL** job with the Foundation license.

  Solution: Perform the procedure in "Discover the SE Oracle Database" in *Discovery and Dependency Mapping Content Guide*.

- **Problem**: When integrating UCMDB with HP Network Node Manager i (NNMi), unable to run the **Layer2 by NNM** job with the Foundation license.

  Solution: For details, see "Run NNMi/Universal CMDB Integration" in *Discovery and Dependency Mapping Content Guide*.

Part II

Introduction

4

Introduction to Discovery and Dependency Mapping

This chapter includes:

Concepts

Tasks

Reference

Discovery and Dependency Mapping – Overview

The Discovery and Dependency Mapping (DDM) process is the mechanism that enables you to collect information about your system by discovering the IT infrastructure resources and their interdependencies. DDM automatically discovers and maps logical application assets in Layers 2 to 7 of the Open System Interconnection (OSI) Model.

DDM discovers resources such as applications, databases, network devices, servers, and so on. DDM also communicates with industry standard or application APIs. Each discovered IT resource is delivered to, and stored in, the configuration management database (CMDB) where the resource is represented as a managed CI.

DDM is an ongoing, automatic process that continuously detects changes that occur in the IT infrastructure and updates the CMDB accordingly. You do not need to install any agents on the devices to be discovered.

Following installation, the network on which the DDM Probe is located, the host on which the Probe resides, and the host's IP address are automatically discovered and a CI is created for each of these objects. These discovered CIs are placed in the CMDB. They act as triggers that activate a DDM job. Every time a job is activated, the job discovers more CIs, which in turn are used as triggers for other jobs. This process continues until the entire IT infrastructure is discovered and mapped.

Once you configure DDM and activate the required patterns, DDM runs on the system, discovers system components, and saves them as CIs in the CMDB. You can discover new objects either manually or automatically. Objects that are outside the Probe's network require additional, manual configuration.

Note: This guide assumes that DDM Probe is installed in the default location, that is, **C:\hp\DDM\DiscoveryProbe\**.

Agentless Technology

DDM is an agentless technology that discovers IT environment components through a dedicated Probe residing on the customer's site. For example, the Netlinks discovery module discovers TCP/IP connections from received NetFlow data.

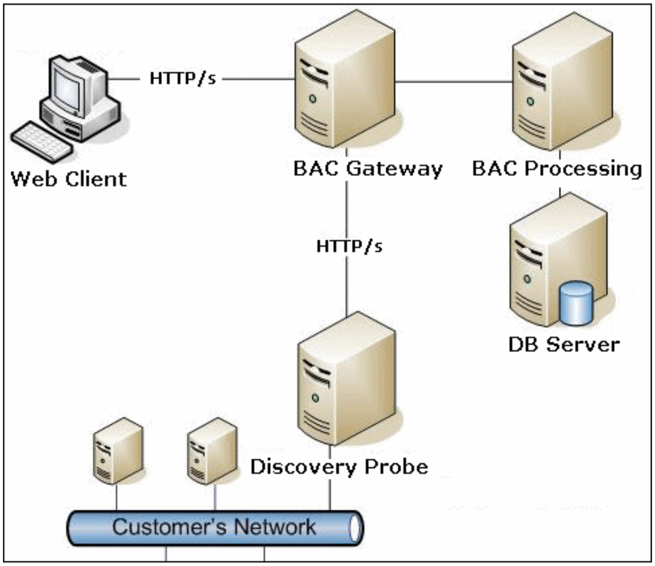
The Probe connects to HP Business Availability Center via http or https traffic to receive new tasks, send task results, and so on. For details on the Probe workflow, see "DDM Probe Tasks" on page 31.

Although DDM is agentless, that is, it does not require the installation of any agent on a customer's machine, it does depend on agents that are already installed such as:

- **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 managerial 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.
- **xCmd.** A remote administration technology similar in functionality to Telnet/SSH that enables launching any console command over Windows machines. xCmd relies on Administrative Shares & Remove Service Administration APIs to function correctly.
- **Application specific.** This agent 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.

Discovery and Dependency Mapping Architecture

Discovery and Dependency Mapping architecture is deployed as follows:



DDM User Interface

- The DDM Probe is the component that performs the data collection and runs on the customer's network.
- The user interface servlet (collectorsUtilities) and the Probe servlets (collectorsServlet, collectorsResultsServlet, and collectorsDownloadServlet) reside on the HP Business Availability Center Gateway Server machine.
- To process clients' requests, these components interact with the HP Business Availability Center Data Processing Server running the DDM components, the viewing system, and the CMDB.

Discovery and Dependency Mapping Components

This section includes the following topics:

- "DDM Probe" on page 53
- "HP Business Availability Center Servers" on page 53
- "Discovery Modules" on page 54
- "Jobs" on page 54
- "Discovery and Dependency Mapping Wizards" on page 55
- "Protocols" on page 55
- "Patterns" on page 55
- "Configuration Files" on page 56
- "External Resources" on page 56
- "Packages" on page 56
- "Scripts" on page 56

DDM Probe

The Probe is the main component responsible for requesting tasks from the server, dispatching them, and sending the results back to the CMDB through the server. You define a range of network addresses for a specific, installed Probe. Each Probe is identified by its name. For details on how the Probe functions, see "DDM Probe Tasks" on page 31.

The DiscoveryProbe.properties file contains configuration parameters. The file is located in **C:\hp\DDM\DiscoveryProbe\root\lib\collectors**. For details, see "The DiscoveryProbe.properties File" on page 39.

HP Business Availability Center Servers

The HP Business Availability Center Gateway Server hosts the servlets that deliver requests to the Probe. The Processing Server receives the results and stores the collected data in the CMDB.

Discovery Modules

The module is a grouping of jobs that logically belong together, can be operated and managed together, and so on. This helps to reduce clutter in the main view when many jobs need to be written, and can also offer better manageability.

When creating a job, you should choose a module for it or create a new module. If you are creating several jobs, the best practice is to split them into logical groups and assign them to modules accordingly.

Jobs

A job enables reuse of a pattern for different DDM processes. Jobs enable scheduling the same pattern differently over different sets of triggered CIs and also supplying different parameters to each set. (To activate DDM, you activate jobs—organized in modules—and not patterns.)

Jobs are organized in modules as follows:

- **Applications.** The modules discover Microsoft Exchange, Oracle E-Business Suite components, the SAP environment based on Computer Center Management System (CCMS), the Siebel environment (such as the Siebel topology and database), WebSphere MQ, and the UDDI registry Web services.
- **Cluster.** The modules discover Microsoft Cluster, ServiceGuard, and Veritas.
- **Database**. DDM first finds instances of databases, then of the database resources (for example, users, tables, tablespaces) for each database instance. Business Availability Center includes predefined default views of the DB2, Oracle, and Microsoft SQL Server databases.
- **Discovery Tools** This module holds the jobs necessary to discover document files and directories, discover hosts, import data from external sources, and serve as a template example.
- **Integration.** These modules are needed for integration between UCMDB and NNM Layer 2 and Storage Essentials.
- **J2EE.** The modules discover JBoss, Oracle Application Server, WebLogic, and WebSphere components.

- **Network.** The modules discover resources on Windows and UNIX hosts, for example, disk information, running processes or services, load balancing, and so on.
- **Virtualization.** The module discovers VMware components.
- **Web Servers.** The modules discover Apache and Microsoft IIS for Windows, SunOne for Solaris, and IBM HTTP Server.

Discovery and Dependency Mapping Wizards

You use one of the DDM wizards (to discover the infrastructure, databases, and J2EE applications) when you need to use the default values set for IP ranges, network credentials, and so on. For details on using a wizard to run DDM, see “Basic Mode Window” on page 119.

Protocols

Discovery of the IT infrastructure components uses protocols such as SNMP, WMI, JMX, Telnet, and so on. For details, see “Domain Credential References” on page 204.

Patterns

Patterns are one of the resources of a Discovery and Dependency Mapping job (DDM job). A pattern includes default configuration parameters, an input TQL (that describes potential input CIs), and scheduling information that define how to perform DDM. A pattern also includes scripts and other code needed for discovery.

A job can either override the default pattern configuration (by associating a specific set of Trigger CIs with each pattern) or can run what is declared in the pattern.

For details on making pattern changes, see “Manage Discovery Resources Window” on page 256. For details on pattern-writing, see Chapter 11, “Content Development and Writing.”

Configuration Files

Configuration files include properties and parameters that are relevant for the DDM patterns. For example, the portNumberToPortName.xml file (that maps a discovered port's number to a port name) includes a list of ports used by DDM when discovering networks. For details on user-definable files, see "Resource Files" on page 235.

External Resources

External resources include all resources external to Business Availability Center that are needed in DDM, for example, a Visual Basic file, a credentials file, and so on.

Packages

Packages contain job definitions, patterns, resources, and tools that enable you to discover IT infrastructure resources such as network extensions, applications, and databases. For details, see "Package Manager" in *Model Management*.

Scripts

Business Availability Center uses Jython scripts for pattern writing. For example, the SNMP_Connection.py script is used by the SNMP_NET_Dis_Connection pattern to try and connect to machines using SNMP. Jython is a language based on Python and powered by Java. For details on pattern-writing, see Chapter 11, "Content Development and Writing."

For details on how to work in Jython, you can refer to these Web sites:

- http://www.jython.org
- http://www.python.org

Discovery and Dependency Mapping Applications

Discovery and Dependency Mapping includes the following applications:

- "Run Discovery" on page 57
- "Set Up Discovery Probes" on page 57
- "Manage Discovery Resources" on page 57
- "Show Status Snapshot" on page 58

Run Discovery

The Run Discovery application enables you to manage the DDM modules and jobs (required for discovering a specific group of CIs). You run the process by activating jobs. You can choose to activate all or some of the jobs in a module. You can also edit jobs, and you can schedule a job to run at a certain time.

For details, see Chapter 6, "Run Discovery."

Set Up Discovery Probes

Set Up Discovery Probes enables you to add Probes to the system and to edit existing Probes. You define the network range that each Probe must cover.

For details, see Chapter 7, "Set Up Discovery Probes."

Manage Discovery Resources

Note: Only users with an advanced knowledge of Discovery and Dependency Mapping should make changes to the resources.

Manage Discovery Resources enables you to view the resources that are needed to perform discovery. You can edit patterns, scripts, configuration files, and you can replace or remove external resources needed in DDM.

For details, see Chapter 8, "Manage Discovery Resources."

Show Status Snapshot

Show Status Snapshot enables you to view details about the scheduling of a particular job as well as job statistics. You can also view the report results in a My BSM portlet.

For details, see Chapter 9, “Show Status Snapshot.”

Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs

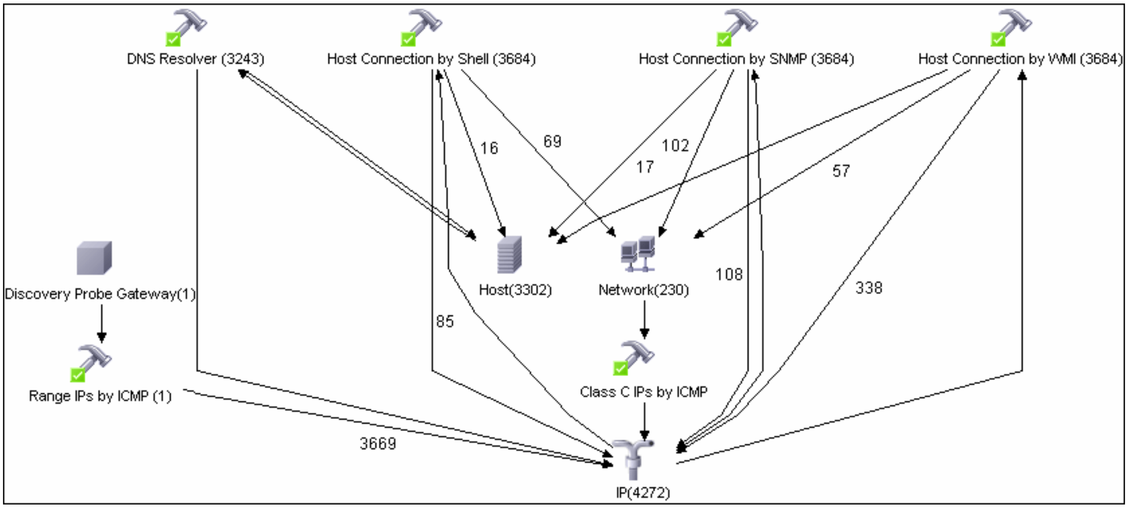
This section describes the functions of Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs. For details on these objects, see “Define Pattern Input (Trigger CIT and Input TQL)” on page 318 and other sections in Chapter 11, “Content Development and Writing.” For details on TQLs, see “Topology Query Language” in *Model Management*.

Trigger CITs

The Trigger CIT defines which CIT is used as the input for a pattern. For example, for a pattern that is going to discover IPs, the input CIT is Network.

Example – Trigger CIT Activating Jobs

The **Network** CIT is a trigger that activates the **Class C IPs by ICMP** job. The **Class C IPs by ICMP** job then discovers instances of IP addresses. These discovered IP addresses themselves become CIs that act as triggers to activate the **Host Connection** jobs, which in turn discover more IP addresses and Network CIs. The process ends when all the IP addresses included in the range defined for the Probe are discovered, as seen in the following illustration:


DNS Resolver (3243)
Host Connection by Shell (3684)
Host Connection by SNMP (3684)
Host Connection by WMI (3684)
16
69
102
17
57
Discovery Probe Gateway(1)
Host(3302)
Network(230)
108
338
85
Range IPs by ICMP (1)
Class C IPs by ICMP
3669
IP(4272)

Trigger CIs

A Trigger CI is a CI in the CMDB that activates a job. Every time a job is activated, the job discovers more CIs, which in turn are used as triggers for other jobs. This process continues until the entire IT infrastructure is discovered and mapped.

For details on adding Trigger CIs to a job, see “Discovery Status Pane” on page 135.

Input TQLs

An Input TQL has two functions:

- **An Input TQL is associated with a pattern.** The Input TQL defines a minimal set of requirements for every Trigger CI included in a job that runs this pattern. (This is true even when no trigger TQL is associated with the job.)

  For example, an input TQL can query for IPs running SNMP, that is, only IPs with installed SNMP agents can trigger this pattern. This prevents the case where a user could manually create a Trigger CI that adds all hosts as triggers to a pattern.

- **An Input TQL defines how to retrieve data information from the CMDB.** Destination data information, even if it is not included in a Trigger CI, can be retrieved by the Input TQL. The Input TQL defines **how** to retrieve the information.

  For example, you can define a relationship between a Trigger CI (a node with the node name of **SOURCE**) and the target CI and then can refer to the target CI according to this node name, in the Triggered CI Data pane. For details, see "Triggered CI Data Pane" on page 269.

  For details on using input TQLs when writing patterns, see "Step 1: Create a Discovery and Dependency Mapping Pattern" on page 317.

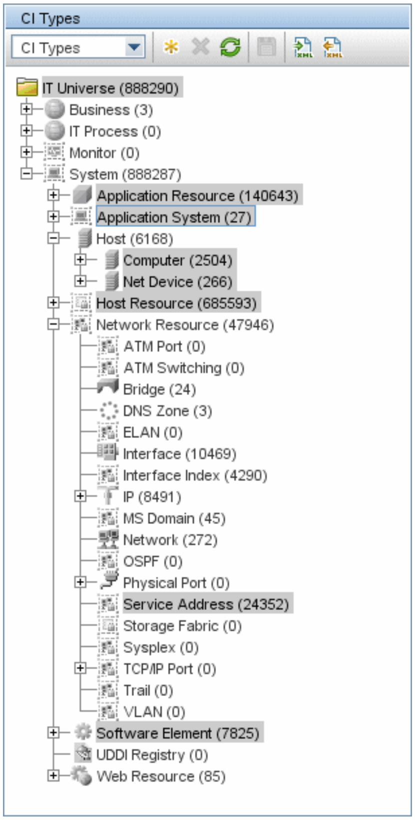
Trigger TQLs

A Trigger TQL associated with a job is a subset of the Input TQL, and defines which specific CIs should be the Trigger CIs for a job. That is, if an Input TQL queries for IPs running SNMP, a Trigger TQL queries for IPs running SNMP in the range 195.0.0.0-195.0.0.10.

Note: A Trigger TQL must refer to the same objects as the Input TQL. For example, if an Input TQL of a pattern queries for IPs running SNMP, you cannot define a Trigger TQL for an associated job to query for IPs connected to a host. This is because some of the IPs may not be connected to an SNMP object, as required by the Input TQL.

Class Model – Overview

This section provides an overview of the UCMDB class model, focusing on the high-level and important CI Types.



CI Types

CI Types

- IT Universe (888290)
  - Business (3)
  - IT Process (0)
  - Monitor (0)
  - System (888287)
    - Application Resource (140643)
    - Application System (27)
    - Host (6168)
      - Computer (2504)
      - Net Device (266)
    - Host Resource (685593)
    - Network Resource (47946)
      - ATM Port (0)
      - ATM Switching (0)
      - Bridge (24)
      - DNS Zone (3)
      - ELAN (0)
      - Interface (10469)
      - Interface Index (4290)
      - IP (8491)
      - MS Domain (45)
      - Network (272)
      - OSPF (0)
      - Physical Port (0)
      - Service Address (24352)
      - Storage Fabric (0)
      - Sysplex (0)
      - TCP/IP Port (0)
      - Trail (0)
      - VLAN (0)
    - Software Element (7825)
    - UDDI Registry (0)
    - Web Resource (85)

This section includes the following topics:

- “The Host CIT” on page 63
- “The Host Resource CIT” on page 63
- “The Software Element CIT” on page 63
- “The Application Resource CIT” on page 64
- “The Application System CIT” on page 65
- “The Service Address CIT” on page 65

The Host CIT

The Host CIT represents any network node (physical or virtual) in your environment.

In UCMDB, there are two host CITs:

- **Computer.** A general purpose device, running general purpose operating systems (for example, UNIX, Windows, mainframe).
- **Network Device.** A network device with a pre-designated purpose (that is, Router, Switch, Load Balancer Device, Firewall, and so on).

Any host in UCMDB is identified either by an IP address or a strong ID value (such as a MAC address or hardware ID). Hosts identified by an IP address are considered incomplete (the **Host Is Complete** attribute has a **false** value). Hosts identified by a strong ID are considered **complete**.

The Host Resource CIT

The Host Resource CIT represents the resources of a host, which include both physical resources (for example, Memory, CPU) and operating system resources (for example, File System, File, Network Share, OS User).

Host Resource CITs are always strongly contained in a Host CIT (they are linked with a container link), that is, in the UCMDB world they cannot exist outside the context of a host.

The Software Element CIT

The Software Element CIT represents a piece of software running on a host that is part of an application solution.

Examples of software elements discovered by DDM include: databases (Oracle, MSSQL, DB2, and Sybase), Web servers, Microsoft Exchange Server, J2EE servers, load balancing software, and cluster software.

Software Element CITs are considered either of a **strong** type or a **weak** type:

- **Strong type Software Element CITs.** These CITs are specialized (they are derived from the Software Element CIT) and include more attributes, different identification rules, a display label, and so on.
- **Weak type Software Element CITs.** These CITs are CI instances of the Software Element CIT itself. They include basic configuration attributes only. Also, only one instance of each software component type can exist in the model. That is, you cannot model two different Oracle instances on a single host using a weak type software element.

A reconciliation mechanism handles the mapping of weak type software elements to their corresponding strong type, once they are discovered. This mechanism enables DDM to detect the existence of software by one method, and report additional details about the software when it is discovered using another method, knowing the two CIs will eventually be merged in the UCMDB.

The reconciliation mechanism relies on a shared attribute value between the weak type and strong type: the Name attribute (**data_name**). All software elements (strong or weak) are created with a distinguishing name: when a strong software element is discovered on a host containing a weak software element with the same name, the weak software element is merged with the strong one.

Weak type software elements are usually discovered by DDM's Host Resources discovery patterns. These patterns rely on a configuration file (Application Signature) that identifies key software according to the running processes and their network interconnections.

Like the Host Resource CIT, a Software Element CIT is strongly contained in a host.

The Application Resource CIT

The Application Resource CIT represents a software element's internal configuration. These CITs are used when attributes are not enough to describe the complex configuration of certain software elements. One group of such resources is the Database Resource group that includes CIs that describe a database's internal configuration (the database schema, database data files, database tables, and so on).

Further generic application resources (for example, Configuration Files, Resource Pool, Web Service) are included in multiple software element CITs, rather than being specific to a single CIT.

The Application Resource CIT is strongly contained in the Software Element CIT.

The Application System CIT

The Application System CIT represents an application deployment. Unlike the Software Element CIT, which is constrained to a one-host context, the Application System CIT is designed to represent application configurations across multiple hosts.

An Application System CI is usually linked via a member relationship to the Software Element CI. The Application CI together with the Software Element CI represent the application in the CMDB.

Examples of Application System CITs include Clusters, SAP System, Siebel Enterprise System, and MS Exchange System.

The Service Address CIT

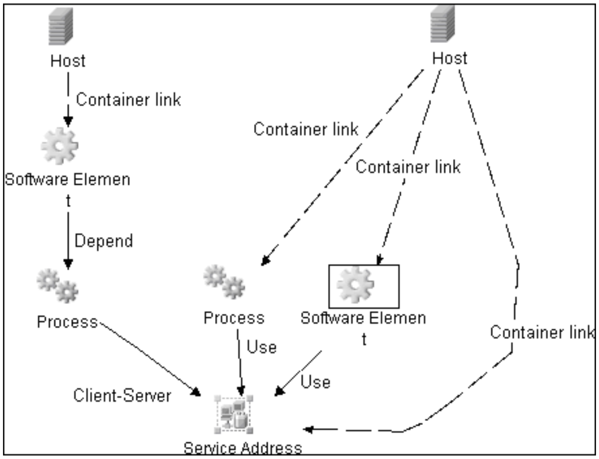
The Service Address CIT derives from the Network Resource CIT and represents an address used by client software to retrieve a service from a software element.

The Service Address CIT key attributes include:

- **Address Type.** Currently acceptable values are **TCP/UDP** or **URL**.
- **Service Address.** The contents of the Service Address field depend on the address type. The TCP/UDP addresses use the format <IP-Address:Port>. The URL addresses contain the accessible URL value.

The mapping of software dependencies relies on the Service Address CIT in the model. The client Software Element is linked via a **depend** relationship to the Client Process CI. The Client Process CI has a Client Server relationship to a remote Service Address CIT. This Service Address CIT is connected through the **use** relationship to the process (Host Resource) and/or to the software element that provides a service at that address.

The Service Address CIT is also strongly contained in the Host CIT.



Class Model Changes

For details on the changes to the class model in version 8.00, see “Class Model Changes” in *Reference Information*. Also, see “Class Model – Overview” on page 62.

DDM Upgrade Information

The following sections include upgrade information:

- “Discovery and Dependency Mapping API Changes” in *Reference Information*.
- “Discovery Modules” in *Reference Information*.
- “Upgrading the DDM DomainScopeDocument File” in *Reference Information*.

Manually Activate a Job

You can activate a job by clicking the **Activate** button in the Discovery Modules pane. You can manually activate a CI by disabling the TQL and adding a CI. (You disable a TQL in the **Edit Probe Limitation for TQL Output** dialog box. You manually add a CI in the Choose CIs to Add dialog box.) The job runs using only the redispatched CIs. For details, see "Discovery Modules Pane" on page 144.

Manually Create a Network CI

A Probe starts by discovering the network on which it is running, so usually there is no need for you to create a network CI. However, if you install the Probe on a certain network, but configure the Probe to discover objects on another network, DDM is not able to discover the network. You must manually create a network CI for the network that the Probe must discover.

To verify that a network CI exists, access the View Manager (**Admin** > **Universal CMDB** > **Modeling** > **View Manager**). Locate the Network folder and verify that the folder contains a Network Topology view.

For details on manually creating a network CI, see "New CI Dialog Box" in *Model Management*.

Schedule Modules to Run

You can set a schedule for a job or a module so that it runs at a certain time. For details, see "Discovery Scheduler Dialog Box" on page 149.

Naming Conventions

When naming entities in DDM, you can use the following characters: a-z, A-Z, 0-9. When entering IP addresses, use only digits and asterisks (*).

Log Files

This section describes the DDM log files and explains how to perform basic troubleshooting. Log files store messages, including errors, relating to the activation of jobs. For details on problem management, see "Managing Problems With Error Reporting" on page 96. For details on setting options for communication logs, see "Execution Options Pane" on page 260.

Severity Levels

Each log is set so that the information it records corresponds to a certain severity threshold. Because the various logs are used to keep track of different information, each is pre-set to an appropriate default level. For details on changing the log level, see "Changing Log Levels" below.

Severity levels are listed here from narrowest to widest scope:

- **Fatal.** This level reports serious errors such as a problem with the infrastructure, missing DLL files, or exceptions.
- **Debug.** This level is used by HP Software Support when troubleshooting problems.
- **Error.** This level reports problems that cause DDM not to retrieve data. Look through these errors as they usually require some action to be taken (for example, to increase time-out, to change a range, to change a parameter, to add another user credential, and so on).
- **Warning.** When a run is successful but there may be non-serious problems that you should be aware of, DDM marks the severity as **Warning**. You should look at these CIs to see whether data is missing, before beginning a more detailed debugging session. **Warning** can include messages about the lack of an installed agent or remote host, or that invalid data caused an attribute not to be properly calculated.
- **Info.** The log records all activity. Most of the information is normally routine and of little use and the log file quickly fills up.
- **Success.** The Trigger CI ran successfully.

Note: The names of the different log levels may vary slightly on different servers and for different procedures. For example, **Info** may be referred to as **Always logged** or **Flow**.

Changing Log Levels

If requested by HP Software Support, you may have to change the severity threshold level in a log (that is, to set verbosity), for example, to a debug level.

To change the severity threshold level:

1. Open the log properties file in a text editor. Log file properties are defined in files in the following directories:

   a. **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\probeMgrLog4j.properties**

   b. **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeGateway\probeGwLog4j.properties**

2. Locate the log parameter. For example:

```
log4j.appender.LOGFILE_Performance.Threshold=DEBUG
```

3. Change the level to the required level. For example:

```
log4j.appender.LOGFILE_Performance.Threshold=INFO
```

   For a description of the log levels, see “Severity Levels” on page 68.

4. Save the file.

   This section includes the following topics:

- “Server Logs” on page 70
- “Probe Logs” on page 71

Server Logs

Server log files reside on the HP Business Availability Center Gateway server. They store information about server activity, including error messages, that occurs on the server side.

The following logs are located in the **<HP Business Availability Center root directory>\log\** or in a sub-folder.

mamAutoDiscovery.log

Contains information about tasks running on the server. The server provides services to the user interface, such as: activating jobs, processing results from the Probe, or creating tasks for the Probe. In a distributed environment, the file resides on the Data Processing server.

Level	Description
Error	All DDM process errors on the server side.
Information	Information about requests being processed.
Debug	Logs mainly for debugging purposes.

Basic Troubleshooting. Check this log when you have invalid user interface responses or errors you need to explore. This log provides information to enable you to analyze the problems.

discoveryServlet.log

This log receives messages from:

- **The Collectors Utilities Servlet.** The user interface connects to the server through this servlet.
- **The Collectors Servlet.** The Probe requests new tasks from the server through this servlet.
- **The Collectors Results Servlet.** The Probe sends new results through this servlet.
- **The Collectors Download Servlet.** The Probe downloads new server data through this servlet.

In a distributed environment, the file resides on the Gateway server.

Level	Description
Error	All errors in the servlet.
Information	Information about user requests and Probe task requests.
Debug	➤ User requests ➤ Probe requests to read DDM tasks. ➤ Probe access of the servlet.

Basic Troubleshooting.

- User Interface–Server communication problems.
- Probe–Server communication problems.

Some processing problems may be written to this log instead of to **mamAutoDiscovery.log**.

mamAutoDiscoveryUpgrade.log

Contains information about the upgrade process.

mamAutoDiscoveryResultsStat.log

Contains the statistics of the results received from the Probe.

Probe Logs

Probe logs store information about job activation that occurs in the Probe Gateway and Probe Manager.

The logs in this section are located in **C:\hp\DDM\DiscoveryProbe\root\logs**.

General Logs

wrapperProbe.log

Records all the Probe's console output in a single log file.

Level	Description
Error	Any error that occurs within the Probe Gateway.
Information	Important information messages, such as the arrival or removal of a new task.
Debug	Record of every Probe access of the servlet.

Basic Troubleshooting. Use this file for any Probe Gateway problems to verify what occurred with the Probe Gateway at any time as well as any important problems it encountered.

probe-error.log

Summary of the errors from the Probe.

Level	Description
Error	All errors in the Probe components.
Information	N/A
Debug	N/A

Basic Troubleshooting. Check this log to verify if errors occurred in the Probe components.

probe-infra.log

List of all infrastructure messages.

Level	Description
Error	All infrastructure errors.
Information	Information about infrastructure actions.
Debug	Messages mainly for debug purposes.

Basic Troubleshooting. Messages from the Probe's infrastructure only.

wrapperLocal.log

Level	**Description**
Error	Any error that occurs within the Probe Manager.
Information	Important information messages such as received tasks, task activation, and the transferring of results.
Debug	N/A

Basic Troubleshooting. Use this file for any Probe Manager problems to verify what occurred with the Probe Manager at any time as well as any important problems it encountered.

Probe Gateway Logs

probeGW-taskResults.log

This log records all the task results sent from the Probe Gateway to the server.

Level	**Description**
Error	N/A
Information	Result details: task ID, job ID, number of CIs to delete or update.
Debug	The **ObjectStateHolderVecto**r results that are sent to the server (in an XML string).

Basic Troubleshooting.

- If there is a problem with the results that reach the server, check this log to see which results were sent to the server by the Probe Gateway.
- The results in this log are written only after they are sent to the server. Before that, the results can be viewed through the Probe JMX console (use the **ProbeGW Results Sender** MBean). You may have to log in to the JMX console with a user name and password.

probeGW-tasks.log

This log records all the tasks received by the Probe Gateway.

Level	Description
Error	N/A
Information	N/A
Debug	The task's XML.

Basic Troubleshooting.

- ➤ If the Probe Gateway tasks are not synchronized with the server tasks, check this log to determine which tasks the Probe Gateway received.
- ➤ You can view the current task's state through the JMX console (use the **Discovery Scheduler** MBean).

Probe Manager Logs

probeMgr-services.log

Java services debug messages.

Level	Description
Error	N/A
Information	N/A
Debug	N/A

Basic Troubleshooting. Check this log to view Java services debug messages.

probeMgr-performance.log

Performance statistics dump, collected every predefined period of time, which includes memory information and thread pool statuses.

Level	Description
Error	N/A
Information	N/A
Debug	N/A

Basic Troubleshooting.

- Check this log to investigate memory issues over time.
- The statistics are logged every 1 minute, by default.

probeMgr-patternsDebug.log

This log contains messages used to debug pattern issues.

Level	Description
Error	N/A
Information	N/A
Debug	N/A

Basic Troubleshooting. Use this log file for debugging patterns.

Troubleshooting and Limitations

For details on using the log files to perform basic troubleshooting, see “Log Files” on page 68.

This section includes the following topics:

Probe Gateway and Probe Manager Activation

Problem. The Probe Gateway or Probe Manager cannot be activated.

Indication. When trying to activate the Probe Gateway or Probe Manager, the console opens and immediately closes.

Verification. To view the exception message, open the following files located in **C:\hp\DDM\DiscoveryProbe\root\logs**:

- For the Probe Gateway: **WrapperProbeGw.log**
- For the Probe Manager: **wrapperLocal.log**

A message is displayed. If one of the following messages is displayed, the problem lies in the memory size definition:

```
Initial heap too small for new size specified.
Incompatible initial and maximum heap sizes specified.
The port number is being used.
```

To solve this problem, see the following Solution. If another message is displayed and you cannot fix the problem, contact HP Software Support.

Solution. There can be several reasons for the activation problem, for example:

- **Inappropriate memory size.** Minimum and maximum memory sizes are allocated for each CMDB component. These definitions are set in the batch.cmd file, under the set memory sizes section. If memory sizes are too high for your workstation, are illegal, or incompatible with one another, you must change them, save the file, and restart the component whose values you changed.
- **Installation path is too long.** If you installed Business Availability Center to a directory with a long path, the operating system or JVM may have problems running the Business Availability Center execution commands. Reinstall Business Availability Center to a different directory that creates a shorter path.

The Probe Gateway and Probe Manager Connection

Problem. The connection between the Probe Gateway and Probe Manager cannot be established.

Indication. The DDM process is not working properly.

Verification. For the Probe Gateway: An error message is displayed in the Probe Gateway log (**WrapperProbeGw.log**, located in **C:\hp\DDM\DiscoveryProbe\root\logs**), as shown in the following example:

```
Failed to connect to probe manager at <server>. Will retry later
```

For the Probe Manager: An error message is displayed in the Probe Manager log (**probe-infra.log**, located in **C:\hp\DDM\DiscoveryProbe\root\logs**), as shown in the following example:

```
Connection attempt to service:jmx:rmi:///jndi/rmi://<Probe GW HOST>:1742/jmxrmi
failed, probe GW may be down
```

Solution. Check the following:

- Verify that the correct port—1742—is defined. The RMI connection port parameter is called appilog.collectors.rmi.port. It is defined in the **DiscoveryProbe.properties** file, located in **C:\hp\DDM\DiscoveryProbe\root\lib\collectors**.
- Verify whether the Probe Manager port is being used by another application. To verify this, in the Windows command interpreter (**cmd.exe**) type: netstat -na. A list of ports that are currently in use is displayed. If the port is in use, either close the other application or change the port number in the **DiscoveryProbe.properties** file.

Host Name Cannot Be Resolved to IP Address

Problem. A host name cannot be resolved to its IP address. If this happens, the host cannot be discovered, and patterns do not run.

Solution. Add the host machine name to the Windows HOSTS file on the Probe machine.

Discovery Tab Missing from Tabs

Problem. The Discovery tab is not displayed in the main page of Business Availability Center.

Solution. Install a license for the Probe. For details, see "Licensing Models for HP Universal CMDB" in the *HP Business Availability Center Deployment Guide* PDF.

Connection Fails

Problem. The connection between the HP Universal CMDB server and the Probe fails due to an RMI or http exception.

Solution. Ensure that none of the Probe ports are in use by another process.

DDM Results Do Not Appear in the Topology Map

Problem. Data that should have been discovered during the DDM process does not appear in the topology map.

Verification. The CMDB cannot retrieve the data or build the TQL results. Check the Statistics Results pane. If the CIs were not created, the problem is occurring during the DDM process.

Solution. Check the error messages in the **probeMgr-services.log** file located in **C:\hp\DDM\DiscoveryProbe\root\logs**.

Networks and IPs

Problem. Not all networks or IPs have been discovered.

Indication. Not all the networks or IPs appear in the topology map results.

Verification. The IP address range in the Set Up Discovery Probes window does not encompass the scope of the networks or IPs that should have been discovered.

Solution. Change the scope of the DDM range:

1 Select **Admin** > **Discovery** > **Set Up Discovery Probes** to open the Set Up Discovery Probes window.

2. Select the Probe and the range.
3. Change the IP address range in the Ranges box as required.

TCP Ports

Problem. Not all TCP ports have been discovered.

Indication. Not all TCP ports appear in the topology map results.

Verification. Open the **portNumberToPortName.xml** file (**Admin** > **Universal CMDB** > **Manage Discovery Resources** > **Network** > **Configuration Files** > **portNumberToPortName.xml**), and search for the missing TCP ports.

Solution. Add the port numbers that should be discovered to the **portNumberToPortName.xml** file.

Status 'Disconnected' for Probe

Problem. Discovery and Dependency Mapping shows a disconnected status for a Probe.

Solution. Check the following on the Probe machine:

- That the Probe is running.
- That there are no network problems.

SSH/Telnet Credentials

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.

SNMP Credentials

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 in the Add Protocol Parameters dialog box. For details, see “Protocol Parameters Dialog Box” on page 202.
- **Solution 3**. Verify that you have the necessary access rights to retrieve data from the MIB objects on the SNMP agent.

Discover Resources on a Windows XP Machine

Problem. Failure to discover resources on a machine running on the Windows platform.

- **Solution 1**. **Start** > **Settings** > **Control Panel** > **System**. In the Remote tab, verify that the following check box is selected: **Allow users to connect remotely to this computer**.
- **Solution 2**. (For Windows XP) In Windows Explorer, select **Tools** > **Folder Options**. In the View tab, clear the **Use simple file sharing (Recommended)** check box.

SAP Discovery Fails

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.

Limitations

- When DDM is installed on a non-English operating system, job and module names are limited to English characters.

- When performing an Oracle Real Application Clusters (Oracle RAC) discovery, note that DDM cannot discover links to the remote machines (the database clients) 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.

5

Class Model Enhancements

Note: This section is relevant for Content Pack 3.00 or later.

This chapter includes:

Concepts

Reference

Class Model Enhancements Overview

For DDM Content Pack 3.00, the class model has been enhanced. You do not need to run any upgrade procedure for these changes in the content packs. For example, changes are limited to adding new CITs or adding new attributes to existing CITs, but not changing key attributes.

The following enhancements have been made to the class model:

- "BIOS UUID Attribute Support" on page 84
- "Software Product Code Attribute Support" on page 87
- "Application Instance Name Attribute Support" on page 89

BIOS UUID Attribute Support

The BIOS UUID attribute (host_biouuid) has been added to the Host CIT. This section explains how DDM supports the discovery of BIOS UUID using different types of protocols.

This section includes the following topics:

- "Windows: NTCMD Protocol" on page 84
- "Windows: WMI Protocol" on page 85
- "UNIX/Sun Solaris/FreeBSD SSH and Telnet Protocols" on page 85
- "The dmidecode Utility Availability" on page 86
- "Troubleshooting and Limitations" on page 86

Windows: NTCMD Protocol

DDM discovers the BIOS UUID data using the **wmic** command.

WMIC query:

```
wmic path win32_ComputerSystemProduct get uuid /format:list <
%SystemRoot%\win.ini
```

Limitation:

If the **wmic** utility is not installed on the remote machine, the attribute is not set. On Windows 2000 and earlier, by default, the **wmic** utility is not installed.

Windows: WMI Protocol

DDM discovers the BIOS UUID data using the **wmic** command.

WMI query:

```
SELECT UUID from Win32_ComputerSystemProduct
```

Limitation:

The WMI class returns the BIOS UUID attribute value with an incorrect order.

UNIX/Sun Solaris/FreeBSD SSH and Telnet Protocols

Discovery is based on DMI data, which can be accessed by the **dmidecode** utility.

Command:

```
dmidecode | grep UUID
```

Limitation:

If the **dmidecode** utility does not exist, DDM cannot retrieve data.

The dmidecode Utility Availability

The **dmidecode** utility is available from:

- Linux i386, x86-64, ia64
- FreeBSD i386, x86-64
- NetBSD i386, x86-64
- OpenBSD i386
- BeOS i386
- Cygwin i386
- Solaris x86

Troubleshooting and Limitations

- The BIOS UUID attribute is available only on UCMDB, versions 8.02, 9.x, and VMware.
- The BIOS UUID attribute may consist of all **0** or **F**, for the following reasons:
  - A machine is a VM and its BIOS UUID attribute is manually overridden or cleared.
  - An internal DMI (system) provider has problems and fails to determine the real UUID. This is the default behavior of DMI.
- The BIOS UUID attribute may be identical on different machines if these machines are VMs and have been copied from one location to another.

Software Product Code Attribute Support

The Software Product Code attribute (software_productcode) has been added to the Software Element CIT. This section explains how DDM supports the discovery of Software Product Code using different types of protocols.

The **ProductCode** property is a unique identifier for a particular product release, represented as a string GUID, for example, "**{12345678-1234-1234-1234-123456789012}**". Letters used in this GUID must be uppercase. This ID must vary for different versions and languages.

A product upgrade that updates a product to an entirely new product must also change the product code. The 32-bit and 64-bit versions of an application's package must be assigned different product codes.

This section includes the following topics:

- "Windows: Shell Protocol" on page 87
- "Windows: WMI Protocol" on page 88

Windows: Shell Protocol

Note: The Product Code attribute is a part of Windows MSI installer, so for UNIX-like distributives, it is not relevant.

To determine the product code, DDM parses the following registry keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{productCode}
```

DDM gains access to the registry through the **reg.exe** or **reg_mam.exe** tool.

The regular expression for determining the product code (in Python notation):

```
"\\Uninstall\\[\{\[\(
]*([\dabcdefABCDEF]{8}(\-[\dabcdefABCDEF]{4}){3}-[\dabcdefABCDEF]{12}).*\n"
```

Windows: WMI Protocol

To determine the product code, DDM parses the following registry keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{productCode}
```

DDM gains access to the registry through the **StdRegProv** class of the \\**root****default WMI** namespace.

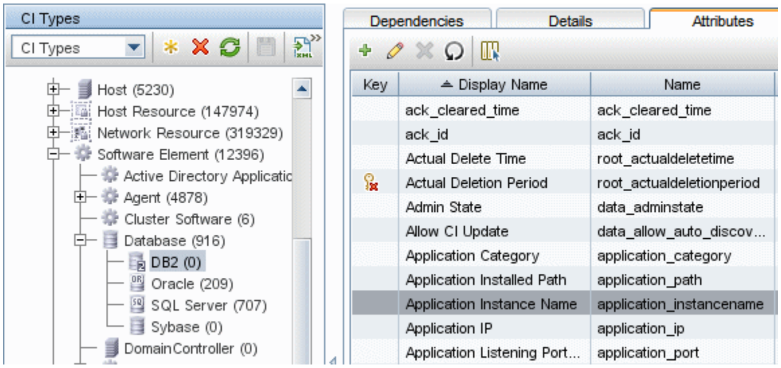
The regular expression for determining the product code (in Python notation):

```
"\\Uninstall\\[\{\[\(
]*([\dabcdefABCDEF]{8}(\-[\dabcdefABCDEF]{4}){3}-[\dabcdefABCDEF]{12}).*\n"
```

Application Instance Name Attribute Support

The Application Instance Name attribute (application_instancename) has been added to the Software Element CIT. This enables DDM to differentiate between instances of the same application on a single machine.

All child Software Element CITs inherit the attribute from the parent (Software Element). During discovery, DDM replaces the value with the appropriate SID. For example, in the J2EE Server application, application_instancename is replaced with j2eeserver_servername; in database applications, application_instancename is replaced with database_dbsid.



- Host (5230)
- Host Resource (147974)
- Network Resource (319329)
- Software Element (12396)
  - Active Directory Applicatic
  - Agent (4878)
  - Cluster Software (6)
  - Database (916)
    - DB2 (0)
    - Oracle (209)
    - SQL Server (707)
    - Sybase (0)
  - DomainController (0)

Key	Display Name	Name
	ack_cleared_time	ack_cleared_time
	ack_id	ack_id
	Actual Delete Time	root_actualdeletetime
🔑	Actual Deletion Period	root_actualdeletionperiod
	Admin State	data_adminstate
	Allow CI Update	data_allow_auto_discov...
	Application Category	application_category
	Application Installed Path	application_path
	Application Instance Name	application_instancename
	Application IP	application_ip
	Application Listening Port...	application_port

Troubleshooting and Limitations

If software is represented by multiple occurrences in the registry, data is retrieved from the first key only.

Part III

Administration

6

Run Discovery

This chapter provides information on using DDM to discover system components.

This chapter includes:

Concepts

Tasks

Reference

Run Discovery – Overview

The Run Discovery pages enable you to activate jobs that discover the components of your system. You activate DDM with one of the following methods:

- Use **Basic Mode** to run DDM for a specific component (for example, the infrastructure, J2EE applications, or databases), using configurable, default preferences.

  For details on the workflow, see “Run Discovery – Basic Mode Workflow” on page 97.

  For details on the Discovery wizard, see “Basic Mode Window” on page 119.

  Note: Basic Mode is displayed by default when you access Run Discovery.

- Use **Advanced Mode** to run DDM to customize a run by making changes to a job.

  For details on the workflow, see “Run Discovery – Advanced Mode Workflow” on page 98.

  For details on the Discovery wizard, see “Advanced Mode Window” on page 118.

  For details on running a specific module, see *Discovery and Dependency Mapping Content Guide*.

Note: To view Help on Run Discovery components:

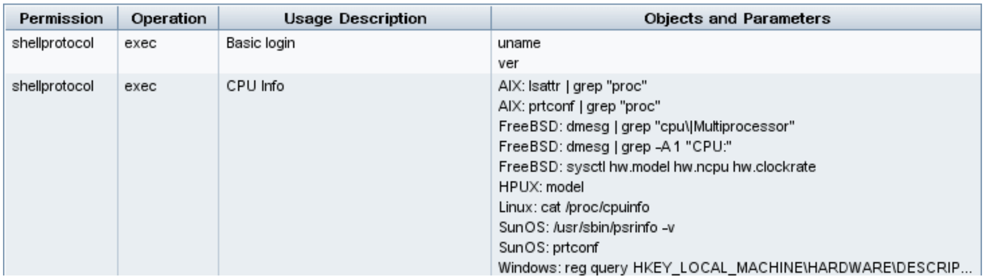
- For details on the Discovery Modules pane, see "Discovery Modules Pane" on page 144.
- For details on the Details tab, see "Details Tab" on page 133.
- For details on the Properties tab, see "Properties Tab" on page 171.
- For details on the Dependency Map tab, see "Dependency Map Tab" on page 131.

Discovery Wizards

As the creation of Discovery wizards entails a very advanced knowledge of DDM, it is recommended that you contact HP Software Support before beginning the work.

Viewing Permissions While Running Jobs

During a job run, you often need to know which credentials are being used to connect to a component in the system. You also often need to know the effect of a run on network performance, for example, whether the job should be run at night instead of during the day. View Permissions enables you to view the objects and parameters of a job's Jython script commands, as can be seen in the following image:



<table>
<tr><th>Permission</th><th>Operation</th><th>Usage Description</th><th>Objects and Parameters</th></tr>
<tr><td>shellprotocol</td><td>exec</td><td>Basic login</td><td>uname
ver</td></tr>
<tr><td>shellprotocol</td><td>exec</td><td>CPU Info</td><td>AIX: lsattr | grep "proc"
AIX: prtconf | grep "proc"
FreeBSD: dmesg | grep "cpu\|Multiprocessor"
FreeBSD: dmesg | grep -A 1 "CPU:"
FreeBSD: sysctl hw.model hw.ncpu hw.clockrate
HPUX: model
Linux: cat /proc/cpuinfo
SunOS: /usr/sbin/psrinfo -v
SunOS: prtconf
Windows: reg query HKEY_LOCAL_MACHINE\HARDWARE\DESCRIP...</td></tr>
</table>

Note: The information you define here is not dynamic, that is, if a pattern is changed, the information in this dialog box is not updated.

For details, see “Discovery Permissions Window” on page 148.

For details on jobs that support this functionality, see “Important Information” on page 148 in the Discovery Permissions window.

Example of Using the Discovery Permissions Window

You are running the Host Connection by Shell job to discover a host running on a UNIX system. An error message in the Discovery Status pane shows that DDM could not access a host through SSH because permission was denied. You display the Discovery Permissions window and see that the command to access the host requires a user with a certain level of permissions. You check the SSH Protocol window and discover that the user defined there does not have that level of permissions.

To resolve the problem, either change the user in the SSH protocol or update the permissions for the existing user in the external system.

Managing Problems With Error Reporting

During DDM, many errors may be uncovered, for example, connection failures, hardware problems, exceptions, time-outs, and so on. DDM displays these errors in Run Discovery, in both Basic and Advanced Mode. You can drill down from the Trigger CI that caused the problem to view the error message itself.

DDM differentiates between errors that can be ignored (for example, an unreachable host) and errors that must be dealt with (for example, credential problems or missing configuration or DLL files). Moreover, DDM reports errors once, even if the same error occurs on successive runs, and reports an error even it if occurs once only.

For details on severity levels, see “Severity Levels” on page 68.

Error Table in Database

All DDM errors are saved to the discovery_problems table in the Probe Manager database schema. (The error information is saved to the database—and is not handled in the Probe's memory—to guarantee delivery to the server.) The Probe holds the latest list of problems for each Trigger CI. After each run, the Probe checks for changes and reports them in the Discovery Status pane. For details, see "Discovery Status Pane" on page 135.

Run Discovery – Basic Mode Workflow

This task describes how to begin mapping your system and its components, using the Discovery wizards. You run this workflow to use default values for the components in an infrastructure, database, or J2EE discovery.

Note: For details of running DDM in Advanced Mode, see "Run Discovery – Advanced Mode Workflow" on page 98.

This task includes the following steps:

- ➤ "Prerequisites" on page 97
- ➤ "Access the Discovery Wizard" on page 97

1 Prerequisites

Verify that the Probe is installed. For details on installing the Probe, see "Install the DDM Probe" on page 18.

For details on licensing, see "Licensing Models for HP Universal CMDB" in the *HP Business Availability Center Deployment Guide* PDF.

2 Access the Discovery Wizard

For details, see the relevant wizard: "Infrastructure Wizard" on page 153, "J2EE Wizard" on page 161, or "Database Wizard" on page 124.

Run Discovery – Advanced Mode Workflow

This task describes how to begin mapping your system and its components. You would use this workflow to customize the components of a module.

Note: For details of running discovery in Basic Mode, see "Run Discovery – Basic Mode Workflow" on page 97.

This task includes the following steps:

- "Prerequisites" on page 98
- "Determine Network Range" on page 98
- "Set Relevant Credentials" on page 99
- "Activate Relevant Jobs" on page 99
- "Make Changes to Relevant Patterns" on page 100
- "Monitor the DDM Process" on page 100
- "View Result Statistics" on page 101
- "Troubleshoot the Results" on page 102

1 Prerequisites

a Verify that the Probe is installed. For details on installing the Probe, see "Install the DDM Probe" on page 18.

For details on licensing, see "Licensing Models for HP Universal CMDB" in the *HP Business Availability Center Deployment Guide* PDF.

b Verify that the relevant packages are deployed.

For details, see Chapter 15, "Package Manager."

2 Determine Network Range

You must define the network range of the network to be discovered. For details, see "Add/Edit IP Range Dialog Box" on page 190.

Note: Patterns try to connect to every IP in a range. Therefore, if a range is wide, network performance may be affected.

3 Set Relevant Credentials

To enable DDM to connect to servers or applications using specific protocols, you must set the relevant credentials (for example, NTCmd, SNMP, TTY, or WMI). For details on protocol parameters, see "Domain Credential References" on page 204. For details on the Details pane in the Set Up Discovery Probes window, see "Details Pane" on page 195.

Note: DDM tries to connect to a host by using each credential in turn. DDM then saves the successful credential. The next time DDM connects to this host, it first tries to connect using the successful credential.

4 Activate Relevant Jobs

Once you have defined the network range and set credentials, you can run discovery on specific jobs. For details, see *Discovery and Dependency Mapping Content Guide*.

Tip: You can view a full description of a job. Select a module and locate the Description pane in the Run Discovery Properties tab, under the DDM pattern name.

Example – Finding SNMP Connections

You can search for all jobs that discover SNMP connections: in the **Run Discovery** > **Discovery Modules** pane, click the **Search for Discovery Job** icon. In the **Find Job**s dialog box, enter **SNMP** in the **Name** box and click **Find All**. For details, see "Find Jobs Dialog Box" on page 152.

5 Make Changes to Relevant Patterns

You can customize patterns to discover infrequent system components. For details on pattern writing, see Chapter 11, "Content Development and Writing."

Important: Do not make changes to default patterns without consulting HP Software Support.

6 Monitor the DDM Process

For details on monitoring the CIs that are discovered by the run, see "Statistics Results Pane" on page 142.

a Define a TQL

You create a TQL query that retrieves information about CIs and CITs from the CMDB. For details, see "Define a TQL Query" in *Model Management*.

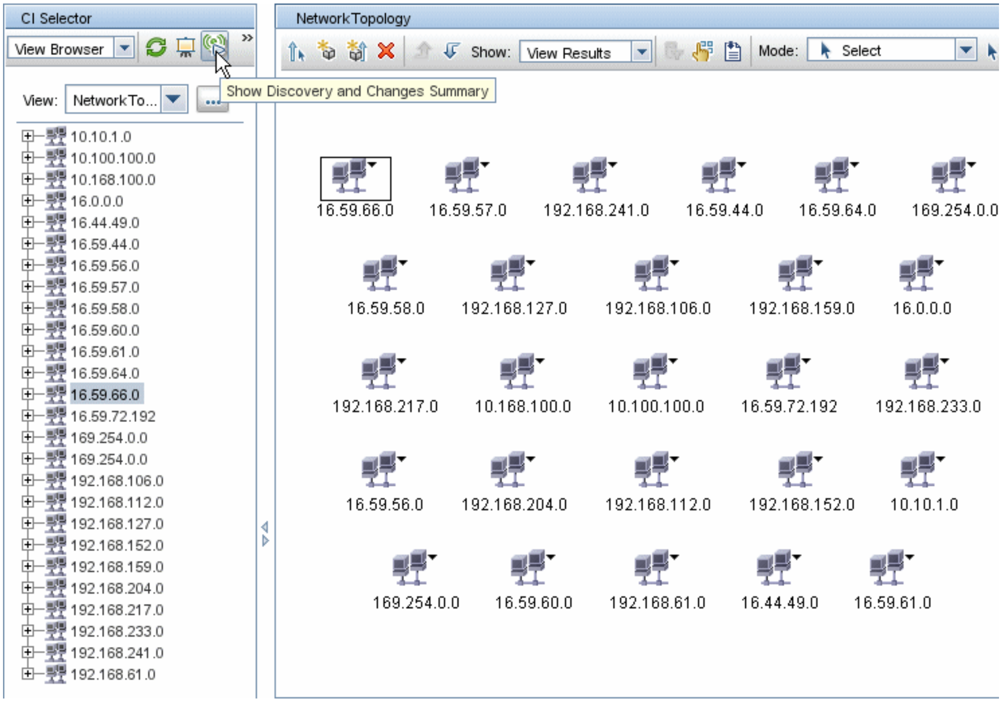
If necessary you can trigger TQLs to manually discover objects. For details, see "Trigger TQLs Pane" on page 176.

b Build a View for each TQL

A view enables you to build a subset of the overall IT universe model, containing only those CIs in the CMDB that relate to a specific discovery. For details, see "View Manager Window" in *Model Management*.

Example – Creating a View to Display Discovered CI Instances

To view the number of instances found by Business Availability Center, select **Admin** > **Universal CMDB** > **Modeling** > **IT Universe Manager**, and display the view you created, as seen in the following illustration:



View Browser

View: NetworkTo...

Show: View Results

Mode: Select

7 View Result Statistics

You can display overall statistics for a job or you can filter the results by time range or by Probe. Each time you log in to Business Availability Center and access Run Discovery, the statistical data is updated so that the data displayed is the latest for the selected module or job.

For details on working with the statistical data, see "Statistics Results Pane" on page 142.

You can view discovered CIs also by accessing the Show Status Snapshot pane. For details, see Chapter 9, "Show Status Snapshot."

8 Troubleshoot the Results

You can check DDM results to see which errors are being reported. For details, see “Manage Errors” on page 102.

Manage Errors

This task describes how to investigate problems that arise during a run.

Note: For details about severity levels and so on, see “Managing Problems With Error Reporting” on page 96.

This task includes the following steps:

- “Prerequisites” on page 102
- “Run the Discovery Wizard or Select the Job” on page 102
- “Locate the Problem CI” on page 102
- “Troubleshoot the Problem” on page 103

1 Prerequisites

Set up DDM. For details, see “Run Discovery – Basic Mode Workflow” on page 97 or “Run Discovery – Advanced Mode Workflow” on page 98.

2 Run the Discovery Wizard or Select the Job

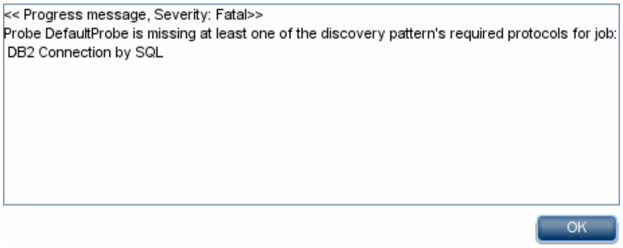
In Basic Mode, you can view error messages for a default job. In Advanced Mode, you can view error messages for one job, one module, or all modules. For details on running a wizard in Basic Mode, see “Run Discovery – Basic Mode Workflow” on page 97. For details on running a job, see “Run Discovery – Advanced Mode Workflow” on page 98.

3 Locate the Problem CI

Use the Discovery Status pane to drill down to the error messages. For details, see “Discovery Status Pane” on page 135.

Example

DDM displays the error message:



```
<< Progress message, Severity: Fatal>>
Probe DefaultProbe is missing at least one of the discovery pattern's required protocols for job:
 DB2 Connection by SQL
```

4 Troubleshoot the Problem

- For Fatal errors, you should contact HP Software Support.
- For other errors, check the CIs. For example, a Trigger CI that does not fall within the Probe’s range may show an error.
- For details on setting communication logs, see “Execution Options Pane” on page 260.
- For details on managing problems, see “Managing Problems With Error Reporting” on page 96.

View Job Information on the DDM Probe

This task describes how to invoke job information (for example, job threads and trigger CIs) saved to the DDM Probe’s MySql database. You work with the JMX console.

This task includes the following steps:

- “Access the MBean Operations” on page 104
- “Locate the Operation to Invoke” on page 104
- “Run the Operation” on page 116

1 Access the MBean Operations

Use the following procedure to access the JMX application on the DDM Probe and to invoke the JMX operations.

a Launch the Web browser and enter the following address:

```
http://<machine name or IP address>.<domain_name>:1977/
```

where <**machine name or IP address**> is the machine on which the DDM Probe is installed. You may have to log in with the user name and password.

b Click the **Local_<machine name or IP address>** > **type=JobsInformation** link.

2 Locate the Operation to Invoke

In the MBean View page, locate the operation:

activateJob

Enter the name of a job and click the button to activate the job immediately. This operation returns a message, for example, <job name> was triggered.

Note: The following message is displayed if the job has not been activated and there is no information about the job in the Probe's database:

Job '<job name>' does not exist in the Jobs Execution table (job was not activated!).

activateJobOnDestination

Enter the name of a job and a Trigger CI and click the button to activate the job immediately on a specific Trigger CI. This operation returns a message, for example, The operation returned with the value: Job <job name> was triggered on destination <CI name>.

Note: Both the **JobID** and **triggerCI** fields are mandatory.

start/stop

These operations start and stop the JobsInformation service. Do not use these operations; instead, restart the Probe itself.

viewJobErrorsSummary

Enter the name of a job to return a list of error messages reported on this job, together with the error severity, the last time that the error was reported, and the number of Trigger CIs that have the error.

For details on the job operation parameters, see "Job Operation Parameters" on page 114.

Click the entry in the **Number of trigger CIs** column to view a list of one job's trigger CIs with errors in the **viewJobTriggeredCIsWithErrorId** page.

viewJobExecHistory

Enter the name of a job to retrieve a history of job invocations. A message is displayed showing the job invocations (the last invocation is shown first).

For details on the job operation parameters, see "Job Operation Parameters" on page 114.

For each invocation the number of Triggered CIs and the total running time is shown. The Execution Details column shows at which times the job was executed. If the Probe shut down in the middle of a job execution and then resumed running or if there were blackout periods during the job execution, several time ranges are shown.

viewJobProblems

Enter the name of a job or the name of a trigger CI to retrieve a list of Trigger CIs that have problems.

Note: You must fill in at least one of the fields.

For details on the job operation parameters, see “Job Operation Parameters” on page 114.

viewJobResultCiInstances

Fill in one or more of the parameters to return a list of CIs that have been discovered by a job.

For details on the job operation parameters, see “Job Operation Parameters” on page 114.

The Object State Holder column displays the code for the CI or relationship defined in the CMDB. For details on creating object state holders for common CITs, see **modeling.py** in “Jython Libraries and Utilities” on page 362. For details on the ObjectStateHolder method, see the *HP Discovery and Dependency Mapping API Reference*.

viewJobResults

Fill in one or more of the parameters to return a list of CIs that have been discovered by a job.

For details on the job operation parameters, see “Job Operation Parameters” on page 114.

When **Hide Touched CIs Info** is set to **True**, the results page displays the following information:

Column	Description
Job Name	Displayed if the **jobID** field is left empty. The job name as it appears in DDM. Click a job to go to its **viewJobStatus** page, to view its status and scheduling information.
CI Type	Click to filter the list to show results for one CIT only.
Total CIs	Click to go to the **viewJobResultCiInstances** page, to view a list of all CIs that have been discovered by a job.
Triggered CIs	Click to go to the **viewJobTriggeredCIs** page, to view a list of all Trigger CIs that have been discovered by a job.
Last Discover Time	The date and time that the job was invoked.

When **Hide Touched CIs Info** is set to **False**, the results page displays the following information:

Column	Description
Job Name	Displayed if the **jobID** field is left empty. The job name as it appears in DDM. Click a job to go to its **viewJobStatus** page, to view its status and scheduling information.
CI Type	Click to filter the list to show results for one CIT only.
Touched CIs	Click to go to the **viewJobResultCiInstances** page, to view a list of those CIs discovered by the job that are **Touched CIs**. For details, see “Job Operation Parameters” on page 114.

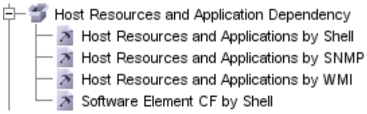
Column	Description
Non Touched CIs	Click to go to the **viewJobResultCiInstances** page, to view a list of those CIs discovered by the job that are not Touched CIs.
Triggered CIs for Touched CIs	Click to go to the **viewJobTriggeredCIs** page, to view a list of those Trigger CIs included in a job that are Touched CIs.
Triggered CIs for Non Touched CIs	Click to go to the **viewJobTriggeredCIs** page, to view a list of those Trigger CIs included in the job that are not Touched CIs.
Last Discover Time	The date and time that the job was invoked.

You can further filter results in the results page by entering text filters in one of the fields, and clicking the **Search** button.

viewJobsStatuses

Click the **viewJobsStatuses** button to return status and scheduling information for all jobs. You can choose to filter the results. For details, see "Job Operation Parameters" on page 114.

The results page displays the following information:

Column	Description
No.	The number of the job in the list.
Job Name	The job name as it appears in DDM, for example:  Host Resources and Application Dependency Host Resources and Applications by Shell Host Resources and Applications by SNMP Host Resources and Applications by WMI Software Element CF by Shell Click a job to go to its **viewJobStatus** page, to view its status and scheduling information.

Column	Description
Status	The severity of the job's status, as calculated by the Probe. **Blocked.** Not in use. **Removed.** The job is no longer active. **Running**. The job is currently running. **Scheduled.** The job is scheduled to run. For details on scheduling jobs, see "Discovery Scheduler Dialog Box" on page 149. A red background signifies that a thread has run longer than expected and may be stuck. A green background signifies that the job is running as expected.
Errors	The number of errors for a specific job. Click to go to the **viewJobErrorsSummary** page, to view a list of error messages reported on this job.
Triggered CIs	The Trigger CIs that have been run by the job. Click to go to the **viewJobTriggeredCIs** page.
Last Invocation	The date and time that the job was last run.
Next Invocation	The date and time that the job is next scheduled to run.
Last Total run duration (seconds)	The total time that it took for the job to run in the last invocation. Compare this result with the average time taken for a job to run. The discrepancy is probably due to periods of time when a job waits for another job to finish.
Avg run duration (seconds)	The average time that the job ran, calculated from all previous invocations.
Recurrence	The number of times that the job was invoked. Click to go to the **viewJobExecHistory** page, to retrieve a history of job invocations.
Results	The number of CITs that have been discovered by the job. Click to go to the **viewJobResults** page to view the CITs.

viewJobStatus

Enter the name of a job to return its status and scheduling information.

For details on the job operation parameters, see "Job Operation Parameters" on page 114.

The results page displays the following information:

Column	Description
Threading info	The total number of worker threads created by the invocation, the free worker threads, and the stuck worker threads.
Total work time	The time that the Probe took to run this job.
Tasks waiting for execution	A list of jobs together with the number of Trigger CIs that are awaiting activation.
Max. threads	The number of threads that are serving this job.
Progress	A summary of the current run, that is, since the specific run was activated. For example, Progress: 2017 / 6851 destinations (29%) means that out of 6851 CIs, 2017 CIs have already run.

Column	**Description**
Working Threads information	**Thread Name.** The thread that is now running this job. Click to go to the **viewJobThreadDump** page. You use this page when a thread is running for a long time, and you must verify that this is because the thread is working hard, and not because there is a problem. **Curr Dest. ID**. The name of the host on which the job is running. **Curr Dest. IP**. The IP for which the job is discovering information. **Work Time (Sec)**. The length of time that this thread is running. **Communication Log.** Click to go to the **viewCommunicationLog** page, to view an XML file that logs the connection between the Probe and a remote machine. For details, see the **Create communication logs** field in the “Execution Options Pane” on page 260.

Column	Description
Discovery Jobs Information table	**Status.** The severity of the job's status, as calculated by the Probe. For details, see "Status" on page 109. **Errors.** Click to go to **viewJobErrorsSummary** page, to view a list of error messages reported on this job. **Triggered CIs.** Click to go to **viewJobTriggeredCIs** page, to view a list of Trigger CIs that are part of a job. **Last invocation.** The date and time that the job was last run. **Next invocation**. The date and time that the job is next scheduled to run. **Last Total run duration (seconds).** For details, see "Last Total run duration (seconds)" on page 109. **Avg run duration (seconds).** For details, see "Avg run duration (seconds)" on page 109. **Recurrence**. The number of times that the job was invoked. Click to go to **viewJobExecHistory** page, to view a history of job invocations.
Results	The number of CITs that have been discovered by the job. Click to go to the **viewJobResults** page to view the CITs.

viewJobTriggeredCIs

Fill in one or more of the parameters to return a list of Trigger CIs that are part of a job.

For details on the job operation parameters, see "Job Operation Parameters" on page 114.

The results page displays the following information:

Column	Description
No.	The number of the job in the list.
Triggered CI ID	The CI instances that have been discovered by the job. Click to go to the **viewJobResults** page to view information about their CITs.
Last Execution	The date and time that the job was last run.
Service Exec. Duration (ms)	The maximum time that it took for a job to run in the last invocation, including periods when the job did not run. Compare this result with the total execution duration. For example, when several jobs run simultaneously, but there is only one CPU, a job might have to wait for another job to finish. The service duration includes this waiting time, whereas the total duration does not.
Total Exec. Duration (ms)	The time that it took for a job to run in the last invocation, excluding the periods when the job did not run.
Last Run Status	The status of the last run, that is, whether the run succeeded or failed. In case of failure, click to go to the **viewJobProblems** page, to view a list of Trigger CIs with problems.
hostID	The name of the machine on which the job is running.
ip_address	The IP address for which the job is collecting information. **Note:** If the Probe has not discovered the ID of the host (for example, because the job is given a range to discover and not a specific IP), information on the host may be missing. In this case, the hostID, ip_address, and ip_domain fields display the string (Empty).
ip_domain	The Probe name as it appears in DDM.

viewJobTriggeredCIsWithErrorId

Note: This operation is part of the inner interface and serves as a helper function. Do not use this page to view Trigger CIs information; instead, use the **viewJobTriggeredCIs** page.

Job Operation Parameters

The following list includes job operation parameters.

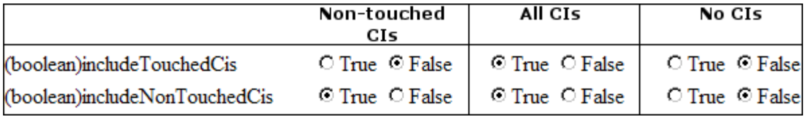
- **ciType**. The name of the CI type (for example, ip, host).
- **data**. A textual field in the DiscoveryResults table that contains information about the discovered object. For example:

```
<object class="ip">
<attribute name="ip_probename" type="String">EBRUTER02</attribute>
<attribute name="ip_address" type="String">16.59.58.200</attribute>
<attribute name="ip_domain" type="String">DefaultDomain</attribute>
</object>
```

- **Error Id.** The error message hash string (error hash ID) that is displayed in the Jobs_Problems table.
- **HideRemovedJobs. True**: do not display jobs that have run previously and are not relevant to the current run.
- **Hide Touched CIs Info**. Touched CIs are CIs which were discovered in previous invocations. DDM already has information about these CIs, so there is no need for the Probe to send the information to the server again. The server identifies that these CIs are relevant and that there is no need to enforce the aging mechanism on them. For details on aging, see “The Aging Mechanism – Overview” in *Model Management*.

  True: the table displays the total number of CIs and the total number of Trigger CIs for each CIT. **False**: The table displays the total number of CIs and Trigger CIs divided between touched CIs and non-touched CIs.

- **includeNonTouched**. Enables filtering the table to view non-touched CIs. Choose between viewing non-touched CIs only, all CIs (touched and non-touched), or none:



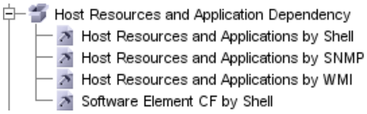
	Non-touched CIs	All CIs	No CIs
(boolean)includeTouchedCis	○ True ● False	● True ○ False	○ True ● False
(boolean)includeNonTouchedCis	● True ○ False	● True ○ False	○ True ● False

- **includeNonTouchedCIs**. See **includeNonTouched**.

- **includeTouched.** Enables filtering the table to view touched CIs. Choose between viewing touched CIs only, all CIs (touched and non-touched), or none.

- **includeTouchedCIs.** See **includeTouched**.

- **jobID.** The name of the job, for example, **Host Resources and Applications by SNMP**:



Host Resources and Application Dependency
- Host Resources and Applications by Shell
- Host Resources and Applications by SNMP
- Host Resources and Applications by WMI
- Software Element CF by Shell

- **maxRows**. The maximum number of rows that should be displayed in the results table. The default is 100 or 1000.

- **maxTriggeredCIs**. See **maxRows**.

- **objectID**. The CMDB object ID.

- **showRemovedJobs**. Shows information about jobs that are not currently scheduled to run, but that have run previously. These jobs take the state of **REMOVED**.

- **showResults**. Indicates whether to display the **Show Results** column. If the Show Results column is present, you can navigate from **viewJobsStatuses** to **viewJobResults**.

- **triggerCI**. The CMDB object ID of the trigger for a job.

- **triggeredCiID**. See **triggerCI**.

3 Run the Operation

- Click the button to run the operation. A message is displayed with the results of the operation run.

  Reload. The number of seconds between automatic reloads of the JMX interface. **0**: The interface is never reloaded. Click the **Reload** button to manually reload the current page (if more operations have been added or removed).

  Unregister. Do not touch (the view becomes inaccessible to the application that is running).

Run Discovery User Interface

This section describes:

- Advanced Mode Window on page 118
- Basic Mode Window on page 119
- Choose CIs to Add Dialog Box on page 121
- Choose Discovery TQL Dialog Box on page 122
- Choose Probe Dialog Box on page 123
- Configuration Item Properties Dialog Box on page 123
- Create New Discovery Job Window on page 123
- Database Wizard on page 124
- Dependency Map Tab on page 131
- Details Tab on page 133
- Discovered CIs Dialog Box on page 144
- Discovery Modules Pane on page 144
- Discovery Permissions Window on page 148
- Discovery Scheduler Dialog Box on page 149
- Edit Probe Limitation for TQL Output Dialog Box on page 151

- Edit Time Template Dialog Box on page 151
- Find Jobs Dialog Box on page 152
- Infrastructure Wizard on page 153
- J2EE Wizard on page 161
- Properties Tab on page 171
- Related CIs Window on page 177
- Show Results for Triggered CI Dialog Box on page 177
- Source CIs Dialog Box on page 178
- Time Templates Dialog Box on page 178
- Triggered CIs Window on page 179
- Trigger TQL Editor Window on page 179

Advanced Mode Window

Description	Enables you to view and manage modules and jobs, to activate jobs, and to follow job progress. Advanced mode includes the following panes: ➤ **Discovery Modules pane.** Each module includes jobs. You activate a module or job to discover a specific group of CIs. For details, see "Discovery Modules Pane" on page 144. **Note**: Basic Mode is displayed by default when accessing Run Discovery. ➤ **Details tab.** Enables you to manage a module's CIs and view CI statistics. For details, see "Details Tab" on page 133. ➤ **Properties tab.** Enables you to view and administer the properties of modules and jobs. For details, see "Properties Tab" on page 171. ➤ **Dependency Map**. Displays a visual representation of the real-time progress of the process. For details, see "Dependency Map Tab" on page 131. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery**.
Important Information	Each change you make in Run Discovery is delivered to and stored in the CMDB. From there, the changes are sent to the Probe. You can verify that changes have been sent to the Probe by opening the **wrapperProbe.log** file located in **C:\hp\DDM\DiscoveryProbe\root\logs\** and searching for the following lines: processing document domainScopeDocument.bin Processing document domainScopeDocument.bin is done. **Note**: Basic Mode is displayed by default when accessing Run Discovery.
Included in Tasks	"Run Discovery – Advanced Mode Workflow" on page 98

Basic Mode Window

Description	Enables you to use a Discovery wizard to discover infrastructure, databases, and J2EE applications. Basic mode includes the following panes: ➤ **List of wizards**. Enables you to choose the wizard to run. For details, see "Infrastructure Wizard" on page 153, "Database Wizard" on page 124, or "J2EE Wizard" on page 161. ➤ **Summary pane**. Enables you to run the wizard and to stop DDM running. For details, see "Summary Pane" on page 120. ➤ **Discovery Progress pane**. Enables you to view a brief run status and to drill down to problematic Trigger CIs. This pane is displayed once discovery has been run for a component. For details on managing errors, see "Discovery Status Pane" on page 135. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery**
Important Information	**Note**: Basic Mode is displayed by default when accessing Run Discovery. For details on Advanced Mode, see "Advanced Mode Window" on page 118.
Included in Tasks	"Run Discovery – Basic Mode Workflow" on page 97
Useful Links	"Run Discovery – Overview" on page 94

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	**Description**
⟳	Click to refresh the list of wizards.

GUI Element (A-Z)	Description
Basic Mode	(Currently displayed) Click to run DDM for a specific component (for example, the infrastructure, J2EE applications, or databases), using configurable, default preferences.
Advanced Mode	Click to run DDM when you need to customize a run by making changes to a job, pattern, and so on. For details, see "Advanced Mode Window" on page 118.

Summary Pane

Description	Enables you to run a Discovery wizard. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery**
Important Information	Depending on whether a wizard has already run, the Summary pane displays the following information: ➤ If a wizard has not yet run, the Summary pane displays the steps to be performed in the wizard and the **Configure and Run** button. ➤ If a wizard has run, the Summary pane displays a summary of the run parameters, the **Configure** and **Stop Discovery** buttons, and the results of the previous run in the Discovery Progress pane. To run a discovery, select a wizard in the left pane and click **Configure** or **Configure and Run** to open the Discovery wizard. To stop a discovery run, click **Stop Discovery**.
Included in Tasks	"Run Discovery – Basic Mode Workflow" on page 97

Choose CIs to Add Dialog Box

Description	Enables you to choose CIs to run with selected jobs. **To access:** ➤ **Discovery** > **Run Discovery**. In the Details tab, locate the **Discovery Status** pane. Click the **Add CI** button. ➤ In the Oracle TNSName File Location page of the Database Wizard, click the **Add CI** button.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Add CI button	**Note:** If you choose CIs with an error status to add to the trigger list, a message is displayed when you click the **Add** button.
Search CIs	Contains filters with which you can limit the number of CIs that appear in the Search Results pane. ➤ **By Discovery TQL**. Select a Discovery TQL to search for those CIs that match the TQL. ➤ **Show only CIs containing.** To search for CIs that include a certain text, enter the text here. ➤ **Exact match.** Select to search for CIs with the exact match of the text label. (By default, you search by entering part of a text. For example, searching for **10** within the IP CIs finds all the IPs that contain 10 in their address. Entering **10** then selecting **Exact match** finds no results.) ➤ **Search.** Click to display the search results.

GUI Element (A–Z)	Description
Search Results	Displays a list of triggered CIs answering to the criteria set in the filter. To add the CIs to the list in the triggered CIs pane, select the CIs. You can make multiple selections. ➤ **CIT**. The CI type of the selected triggered CI. ➤ **CI**. The label of the triggered CI. ➤ **Related Host**. The label for the host related to the triggered CI. ➤ **Related IPs**. The IPs of the related host. **Page.** The list of CIs is divided into pages. The number in the Page box indicates which page is currently displayed. To view other pages, use the up and down arrows, or type the page number, and press **Enter**. To determine the number of CIs that appear on a page, right-click either the up or down button and choose the required number. The default is 25.

Choose Discovery TQL Dialog Box

Description	Enables you to add a trigger TQL to a job. **To access:** Click the **Add TQL** button in the Trigger TQLs pane.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<Discovery TQL name>	The TQLs that can query the CMDB for the selected CIT.
TQL Preview	Hold the cursor over an element to view details.

Choose Probe Dialog Box

Description	Enables you to filter the Probe list. **To access:** Click a Filter button in the **Run Discovery** > **Details** tab: ➤ **Triggered CIs** pane Filter button. For details on the menu options, see "Discovery Status Pane" on page 135. ➤ **Statistics** pane Filter button. For details on the menu options, see "Statistics Results Pane" on page 142.

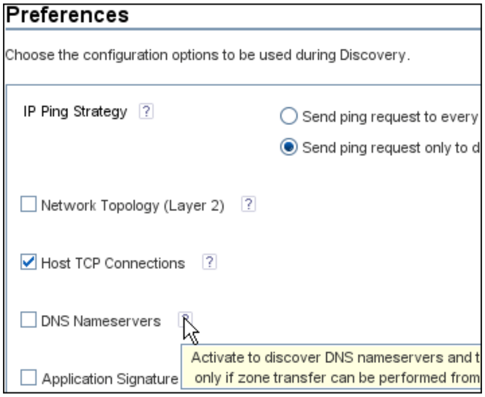
Configuration Item Properties Dialog Box

Description	Enables you to view CI properties. **To access:** In the **Discovered CIs** dialog box, right-click a CI and choose **Properties**.
Important Information	For details, see "Configuration Item Properties Dialog Box" in *Model Management*.

Create New Discovery Job Window

Description	Enables you to create a job. **To access:** Right-click a module in the Discovery Modules pane, and choose **Create New Job**.
Important Information	For details on the panes in this window, see: ➤ "Discovery Job Details Pane" on page 134 ➤ "Parameters Pane" on page 175 ➤ "Trigger TQLs Pane" on page 176 ➤ "Global Configuration Files Pane" on page 267 ➤ "Discovery Scheduler Pane" on page 171

Database Wizard

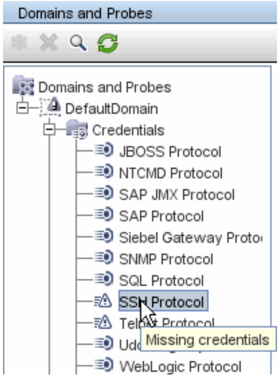
Description	Enables you to discover databases such as DB2, Oracle, Microsoft SQL, and Sybase. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery** > **Basic Mode**. Select the Database wizard from the list in the left pane. Click **Configure and Run**.
Important Information	For more information, hold the pointer over a question mark icon:  Preferences Choose the configuration options to be used during Discovery. IP Ping Strategy ? ○ Send ping request to every ● Send ping request only to d □ Network Topology (Layer 2) ? ☑ Host TCP Connections ? □ DNS Nameservers ? Activate to discover DNS nameservers and t only if zone transfer can be performed from □ Application Signature
Wizard Map	The Database Discovery wizard contains: **Database Wizard** > Define Credentials > Database Port Scanning > Custom JDBC Drivers > Oracle TNSName File Location > Schedule Discovery > Summary

Define Credentials

Description	Enables you to configure connection data for each protocol.
Important Information	➤ You configure protocols depending on what must be discovered and which protocols are supported on your site’s network. ➤ For a list of protocols, see “Domain Credential References” on page 204. ➤ General information about the wizard is available in “Database Wizard” on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > **Define Credentials** > Database Port Scanning > Custom JDBC Drivers > Oracle TNSName File Location > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Add new connection details for selected protocol type.
❌	Remove a protocol.
✎	Edit a protocol. Click to open the Protocol Parameters dialog box.

GUI Element (A–Z)	Description
⇧ ⇩	Move a protocol up or down. DDM executes all the protocols in the list with the first protocol taking priority.
Protocol	Click to view details on the protocol, including user credentials. **Note**: A missing credential is represented by an icon ⚠, as shown in the following image:  Domains and Probes Domains and Probes DefaultDomain Credentials JBOSS Protocol NTCMD Protocol SAP JMX Protocol SAP Protocol Siebel Gateway Proto SNMP Protocol SQL Protocol SSH Protocol Tel Protocol Udc Missing credentials WebLogic Protocol

Database Port Scanning

Description	Enables you to discover the port itself and subsequently to discover the database.
Important Information	General information about the wizard is available in "Database Wizard" on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > Define Credentials > **Database Port Scanning** > Custom JDBC Drivers > Oracle TNSName File Location > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to add a port to the port list. The Add New Port dialog box opens. Select the ports and click **OK**. To edit existing system ports, in the Add New Port dialog box, click **Edit Known System Ports**. The Edit Known System Ports dialog box opens. Select the port and click the **Edit** button. In the dialog box that opens, make changes to the entries and click **OK**. To add a port to the list, in the Edit Known System Ports dialog box, click the **Add** button. Enter details of the port name, number and type and click **OK**.
❌	Select a port and click the button to remove the port from the list.

Custom JDBC Drivers

Description	Enables you to select the JAR file for the DB2 and Sybase JDBC drivers.
Important Information	General information about the wizard is available in “Database Wizard” on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > Define Credentials > Database Port Scanning > **Custom JDBC Drivers** > Oracle TNSName File Location > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
DB2 JDBC Driver	Select the check box and click **Import file...** to locate the appropriate JAR file in the DB2 JDBC installation, as follows: ➤ db2java.zip ➤ db2jcc.jar ➤ db2jcc_license_cu.jar ➤ db2jcc_license_cisuz.jar
Sybase JDBC Driver	Select the check box and click **Import file...** to locate the jconnectXXX.jar JAR file in the Sybase JDBC installation.

Oracle TNSName File Location

Description	Enables the discovery of Oracle databases. You provide the location of the TNSNames.ora configuration file that contains database information needed to discover Oracle databases, such as port, host, SID, and so on.
Important Information	General information about the wizard is available in "Database Wizard" on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > Define Credentials > Database Port Scanning > Custom JDBC Drivers > **Oracle TNSName File Location** > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Server Host	Select the hosts on which the TNSNames.ora file is located. Click the **Add CI** button to choose the Trigger CIs that represent these hosts. For details, see “Choose CIs to Add Dialog Box” on page 121. ➤ **CIT**. The CI type of the selected triggered CI. ➤ **CI**. The label of the triggered CI. ➤ **Related Host**. The label for the host related to the trigger CI. ➤ **Related IPs**. The IPs of the related host.
TNSNames.ora file location	Enter the location of the TNSNames.ora file in the server host system. You can enter several locations (separate the locations by commas). If you terminate the path with a delimiter (for example, c:\temp\), DDM assumes that the file name is **tnsnames.ora**.

Schedule Discovery

Description	Enables you to define a schedule for a specific job.
Important Information	General information about the wizard is available in “Database Wizard” on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > Define Credentials > Database Port Scanning > Custom JDBC Drivers > Oracle TNSName File Location > **Schedule Discovery** > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✎	You define a time template in the Discovery Scheduler pane of the Properties tab. For details, see "Discovery Scheduler Pane" on page 171.
Allow Discovery to run at	Choose the time at which the job should run.
Repeat Every	Select how often the job should run.

Summary

Description	Enables you to review the wizard definitions before running a discovery.
Important Information	To make changes to the run, click the **Back** button. General information about the wizard is available in "Database Wizard" on page 124.
Wizard Map	The Database Discovery wizard contains: Database Wizard > Define Credentials > Database Port Scanning > Custom JDBC Drivers > Oracle TNSName File Location > Schedule Discovery > **Summary**

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Run	Click the button to run a discovery.

Dependency Map Tab

Description	Displays a visual representation of the real-time progress of the discovery process. The map displays: ➤ CIs that were triggered by a job ➤ CIs that were discovered as a result of the activated job. **To access:** Click the **Dependency Map** tab in the Run Discovery window.
Important Information	Depending which level you select in the Discovery Modules pane, different information is displayed in the Dependency Map tab. If you select: ➤ **The Discovery Modules root**, and select the **Show only active Discovery jobs** check box, the Dependency Map displays only active jobs and their interdependencies. ➤ **The Discovery Modules root**, and clear the **Show only active Discovery jobs** check box, the Dependency Map displays all Discovery jobs and their interdependencies. ➤ **A module**, a topology map is displayed showing the module's active and inactive jobs. ➤ **A job**, the topology map highlights the job in the module's map.
Useful Links	"Discovered CIs Dialog Box" on page 144

The following elements are included (unlabeled GUI elements are shown in angle brackets):

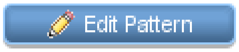
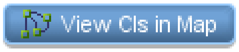
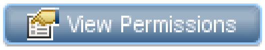
UI Elements (A-Z)	Description
<right-click menu>	Use the right-click menu to view details for a job, CI, or link, for example, the number of CI instances (of a specific type) in the CMDB or the number of CI instances created by a specific job. Depending on which object is selected, the following menu options are displayed: ➤ When a job is selected: **Show discovered CIs.** Click to view the CIs discovered by the job. To filter the query, select a CIT from the menu. **Show trigger CIs.** Click to view the CIs that triggered the job. ➤ When a CI is selected: **Show all CIT instances.** Click to view all CIs of this CI type. ➤ When a link from a CI to a job is selected: **Show trigger CIs for job.** Click to view CIs (of the selected type) that triggered the job. ➤ When a link from a job to a CI is selected: **Show discovered instances.** Click to view CIs (of the selected type) that were discovered by the job.
<Toolbar>	For details, see "Toolbar Options" in *Model Management*.
<Tooltip>	Hold the pointer over a CI or job to display a description.
Show only active Discovery jobs	When the Discovery Modules root is selected in the Discovery Modules pane, this check box is displayed. Select to display all active jobs (from any module).

Details Tab

Description	Enables you to view and administer modules and jobs, to follow the progress of the DDM process, and to manage errors during discovery. **To access:** Click the **Details** tab in Run Discovery.
Important Information	Depending which level you select in the Discovery Modules pane, different information is displayed in the Details tab. If you select: ➤ **The Discovery Modules root** or a **Discovery module**, the Discovery Status and Statistics Results panes are displayed with information and statistics about all active jobs and errors discovered during a run. For details, see “Discovery Status Pane” on page 135 and “Statistics Results Pane” on page 142. ➤ **A job**, the Discovery Job Details, Discovery Status, and Statistics Results panes are displayed. For details, see “Discovery Job Details Pane” on page 134, “Discovery Status Pane” on page 135, and “Statistics Results Pane” on page 142. ➤ Several jobs or modules, the Selected Items pane is displayed. For details, see “Selected Items Pane” on page 141.
Included in Tasks	“Managing Problems With Error Reporting” on page 96

Discovery Job Details Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
	Click to go to the pattern in the Discovery Resources window.
	You can choose to view a map of the CIs and links that are discovered by the pattern, instead of a list. Click the button to open the Discovered Classes Map window. The selected pattern is shown together with its CIs and relationships. Hold the cursor over a CIT to read a description in a tooltip.
	Click to view permissions that are defined for specific patterns. For details, see “Discovery Permissions Window” on page 148 For details on editing these permissions, see “Permission Editor Dialog Box” on page 271.
Discovery Pattern	The pattern needed by the job to discover the CIs.
Discovered CIs	The CIs that are discovered by this job.
Input CI Type	The CIT that triggers the CIs for this job.
Job Name	The name and description of the job and the package in which it is located.

Discovery Status Pane

Description	Enables you to view a run status and to drill down to problematic Trigger CIs, to uncover specific problems that DDM encountered during the run, for example, incorrect credentials. In **Basic Mode**, enables you to view the results of the previous run for the selected job type (that is, infrastructure, database, or J2EE application). In **Advanced Mode**, enables you to view the results of the previous run for a selected module or job, or for all modules. **To access:** ➤ In Basic Mode, locate the Discovery Overview pane. ➤ In Advanced Mode, select a module or job, click the **Details** tab, and locate the Discovery Status pane.
Important Information	➤ You can use the SHIFT and CTRL keys to select adjacent and non-adjacent CIs in a list. ➤ Depending which level you select in Advanced Mode in the Discovery Modules pane, information is displayed in the Discovery Status pane for all modules, for a specific module, or for a specific job. ➤ The information in this pane is automatically refreshed every thirty seconds.
Included in Tasks	"Manage Errors" on page 102 "Check Status of Application Discovery (Rediscover a View)" in *Model Management*.
Useful Links	"Managing Problems With Error Reporting" on page 96

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
⇧	Click to return to the upper pane.
⤵	Click to drill down to the Trigger CI that includes the problem. **Note**: This icon is displayed only when you can drill down from error or warning links.
⟳	Click to refresh the status view.
➕	Click to add a newly-discovered CI. For details, see “Choose CIs to Add Dialog Box” on page 121.
✖	Click to remove a CI from the list, if the CI is no longer of interest. The CI is deleted from the specific job.

GUI Element (A-Z)	Description
⏷	Click and choose an option from the menu: ➤ **By Status.** Displays a list of Trigger CIs: ➤ **All.** Displays all the Trigger CIs. ➤ **Waiting for Probe.** Displays the Trigger CIs that are ready to be dispatched and are waiting for the Probe to retrieve them. ➤ **In Progress.** Displays the Trigger CIs that are active and are running on the Probe. ➤ **In progress (being removed).** Displays the Trigger CIs that are being removed from the **Trigger CIs** list. ➤ **Success**, **Failed**, **Warning**. Displays only those CIs that have the selected status. ➤ **By Probe.** Displays only the CIs triggered by a selected Probe. Click to open the Choose Probe dialog box. ➤ **By Dispatch Type.** Displays a list of CIs according to one of the following options: ➤ **All**. Displays both CIs that are used to manually activate the job and Discovery TQLs that are used to automatically activate the job. ➤ **Manually added.** Displays the CIs that are used to manually activate the job. ➤ **By Discovery TQL**. Displays the CIs that are used to automatically activate the job. ➤ **Reset.** Click to remove any filters.
💬	Click to display a message box containing an explanation of the failure. (You can also view messages by right-clicking the CI and selecting **Show error details**.)
☞	Click to open the Triggered CIs dialog box with additional information about the CI. For details, see “Triggered CIs Window” on page 179.
📊	Show statistics for a specific Trigger CI.
⟳	Click to rerun the discovery.
🔍	Find a CI.

GUI Element (A-Z)	Description
<drill down>	You can drill down from a job or a module. ➤ Drill down from a job to view a list of Trigger CIs that are included in the job. ➤ Drill down from a module to view a list of the jobs in the module and the number of CIs returned by each job. Drill down from a job to its Trigger CIs. **Note:** A Trigger CI can be present in more than one job.

GUI Element (A-Z)	Description
<right-click CI menu>	Right-click a CI to: ➤ **Show error details.** Displays a list of the different types of errors returned by this CI. For details on severity, see “Severity Levels” on page 68. ➤ **Remove.** Select to delete the CI from the job. The CI is removed from that job only, even if it appears in more than one job. ➤ **Rerun Discovery.** To run a specific CI or set of CIs, select the CIs. They are added to the list of CIs that the Probe is going to run (**Waiting for Probe**). ➤ **Show results for triggered CIs.** DDM sends an ad-hoc request to the Probe and retrieves the latest results of the job (CIT name and number of discovered CIs) that is running on a specific trigger CI. This ad-hoc request does not run the job, but brings the results of the previous job run that are stored in the Probe’s database. If the job has not yet run for this trigger CI, a message is displayed. See “Show Results for Triggered CI Dialog Box” on page 177. If no communication log exists in the Probe, a message is displayed. You can choose that DDM always creates communication logs. For details, see “Execution Options Pane” on page 260. ➤ **Debug.** Choose between: ➤ **View communication log for triggered CI.** Opens the log that includes information about the connection between the Probe and the remote machine. This is on condition that you have set the **Create communication log** to either **Always** or **On failure**. For details, see “Execution Options Pane” on page 260. ➤ **Go to pattern.** Displays the pattern that is included in the job in Manage Discovery Resources. ➤ **Go to job.** Displays the job in which the CI is included. ➤ **Edit script**. Select a script to open it in a script editor. ➤ **Undispatch.** Removes the Trigger CI.

GUI Element (A-Z)	Description
Failed	Displays those CIs that returned a severity of type Error or Fatal. Right-click a job to rerun discovery. Double-click a job to display the error message. Right-click an error to deactivate or rerun a job.
In progress	Displays the number of Trigger CIs that are awaiting their turn to be run. Click to view the jobs that are waiting to be run.
Look for	To search for a specific Probe, related host, or related IP, enter part of its name in the box and click **Search**.
Progress	The indicator shows a summary of the current discovery run, that is, since the specific run was activated.
Success	DDM displays the number of CIs that have been run successfully, that is, without errors. Click to view the jobs (and the number of CIs in each job) that completed successfully. Select a CI and use the right-click CI menu to view information. **With warnings.** Click to view a warning message for each job. Double-click a message to view the CIs that finished successfully with a warning. Right-click a message to view the right-click CI menu.
Total	Displays the status of all of a job's Trigger CIs. Double click a **Warning** or **Error** status to open the Message dialog box.
Waiting for Probe	The Trigger CIs that are either waiting for the Probe or are waiting to run.

Selected Items Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<right-click menu>	**Edit Scheduling**. Click to open the Discovery Scheduler to define a schedule for a specific job. For details, see "Discovery Scheduler Pane" on page 171.
invoke immediately	➤ A check mark signifies that the Discovery job runs as soon as the triggered CI reaches the Probe. In this case, the Invoke on new triggered CIs immediately check box is selected in the Properties tab. ➤ If this column does not contain a check mark, the job runs according to the schedule defined in the Schedule Manager.
Job name	The name of the job.
Schedule info	The scheduling information of the job as defined in the Discovery Scheduler.
Trigger TQLs	The name of the TQL that activated the job. For details, see "Trigger TQLs Pane" on page 176.

Statistics Results Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
	Select a CI and click the View Instances button to view CI instances and their attributes. The Discovered CIs dialog box opens. Under the following conditions, a message is displayed: ➤ All the CIs that were discovered by this job were already discovered by another job. ➤ All the CIs that this job discovered have been deleted. ➤ The CI instances were discovered in a previous version. (In version 7.0, you cannot view instances of CIs discovered in a previous version.) **Note**: ➤ You can also view CI instances by double-clicking a row. ➤ CITs with no instantiated instances are displayed.
	Select the time range or Probe for which to display statistics about the CITs. ➤ **By Time Range**: ➤ **All.** Displays statistics for all job runs. ➤ **From Now/Last Minute/Hour/Day/Week**. Choose a period of time for which to display statistics about the CITs. ➤ **Custom Range.** Click to open the Customize Statistics Time Range dialog box. Enter the date or click the arrow to choose a date and time from the calendar, for the **To** and **From** dates. To delete a date, click **Reset**. ➤ **By Probe**: To view statistics for a specific Probe, select to open the Choose Probe dialog box.

GUI Element (A–Z)	Description
🔄	Click to retrieve the latest data from the server (job results are not automatically updated in the Statistics pane).
<**Column title**>	Click a column title to change the order of the CITs from ascending to descending order, or vice versa.
<**right-click a title**>	Choose from the following options: ➤ **Hide Column.** Select to hide a specific column. ➤ **Show All Columns.** Displayed when a column is hidden. ➤ **Customize.** Select to display or hide columns and to change the order of the columns in the table. Opens the Columns dialog box. ➤ **Auto-resize Column.** Select to change a column width to fit the contents. For details, see “Select Columns Dialog Box” in *Reference Information*.
CIT	The name of the discovered CIT.
Created	The number of CIT instances created in the period selected or for the selected Probe.
Deleted	The number of CIT instances deleted in the period selected or for the selected Probe.
Discovered CIs	The number of CIs that were discovered for each CI type.
Filter	The time range set with the Set Time Range button.
Last updated	The date and time that the statistics table was last updated for a particular job.
Total	The total number of CIs in each column.
Updated	The number of CIT instances that were updated in the period selected.

Discovered CIs Dialog Box

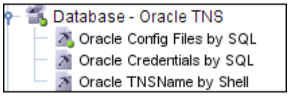
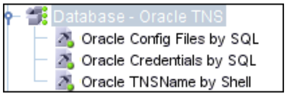
Description	Enables you to view CI instances of a CIT discovered by a job. **To access:** ➤ In the Statistics Results pane, select a CIT, and click the **View instances** button. ➤ In the Dependency Map tab, select **Show Discovered CIs** or **Show all instances**.
Important Information	➤ The Discovered by <job name> window includes the same information as the Element Instances window. For details, see "Element Instances Dialog Box" in *Model Management*. ➤ Depending on whether you select **Show discovered CIs** or **Show all instances** in the Dependency Map, you can view either all CIs discovered by a selected job or all CIs of a selected type.

Discovery Modules Pane

Description	Enables you to view and manage modules and jobs. Each module includes the jobs necessary to discover specific CIs. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery**. The default view is called **Basic Mode** and displays the Discovery Wizard. You can run the J2EE, database, or infrastructure discovery. Click **Advanced Mode** to view all modules.
Important Information	**Caution:** Only administrators with an expert knowledge of the DDM process should delete modules. **Obsolete.** Contains several modules that are no longer relevant but remain for backward compatibility and upgrade purposes. Do not use these modules on new installations. **No module**. Contains jobs that are not included in any other module.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
⟳	**Refresh All.** Updates the modules.
🔍	**Find Job.** Click to open the Find Jobs dialog box. For example, to search for all jobs that discover SNMP connections, click the Filter icon. In the **Find Jobs** dialog box, enter **SNMP** in the **Name** box and click **Find All**. For details, see “Find Jobs Dialog Box” on page 152.
Activate	You can run one job or several jobs in a module, and one or several modules. Select either the jobs or modules and click **Activate**.
Deactivate	Select the jobs or modules to be stopped and click **Deactivate**.
	Represents the module root. To create a module, right-click to enter the name of the module you are creating. **Note:** A name is case sensitive. Names beginning with an upper case letter appear in the Discovery Modules list before names beginning with a lower case letter.
	Represents a module.
	Represents a job. Click to display information about the job. To view a pattern description, hold the pointer over a job. Jobs contain configuration information derived from patterns and other resources and are the entities controlled by users, for example, when activating or deactivating a module. For details on the right-click menu, see “Right-Click Menu” on page 147.

GUI Element (A–Z)	Description
●	One green dot signifies that some of a module's jobs are activated:  Database - Oracle TNS Oracle Config Files by SQL Oracle Credentials by SQL Oracle TNSName by Shell
●●●	Three green dots signify that all of a module's jobs are activated:  Database - Oracle TNS Oracle Config Files by SQL Oracle Credentials by SQL Oracle TNSName by Shell
!	An exclamation mark signifies that one or more of the jobs is experiencing a problem that could affect the DDM process, for example, a protocol connection failure. To view the reason for the problem, click the **(show errors)** link in the Discovery Status pane. For details, see "Failed" on page 140. **Note**: If a problem is resolved by clicking the **Refresh All** button, the Problem Indicator disappears.
Basic Mode	Click to run DDM for a specific component (for example, the infrastructure, J2EE applications, or databases), using configurable, default preferences. For details, see "Basic Mode Window" on page 119.
Advanced Mode	(Currently displayed) Click to run DDM to customize a run by making changes to a job, pattern, and so on.

Right-Click Menu

GUI Element (A–Z)	Description
Activate	Click a module to run all its jobs. To run a specific job, select and activate it. The Discovery Module discovers CITs and relationships of the types that are described in each job, and places them in the CMDB. For example, the Class C IPs by ICMP job discovers the Depend, IP, and Member CITs and relationships.
Create New Job	Click to open the Create New Discovery Job. For details, see "Create New Discovery Job Window" on page 123.
Create New Module	Click to define a new name for the module root.
Deactivate	Stop the module or job from running.
Delete	Click and answer **Yes** to the warning message.
Delete job	Click and answer **Yes** to the warning message.
Go to Pattern	Click to edit the pattern in the Manage Discovery Resources window.
Edit Scheduling	Click to open the Discovery Scheduler to define a schedule for a specific job.
Rename job	Click to open the Choose Name dialog box. Enter a new name for the job. **Note:** You cannot rename active jobs.
Run Now	Click to run the job again using the selected Trigger CIs.
Save as...	Click to clone the job.

Discovery Permissions Window

Description	Enables you to view permissions data for jobs. **To access: Run Discovery** > **Advanced Mode**. Select a job. Locate the **Discovery Job Details** pane in the **Details** tab. Click the **View Permissions** button.
Important Information	In version 8.01 or later, you can view permissions data for the following jobs: ➤ Host Connection by Shell ➤ Host Connection by SNMP ➤ Host Connection by WMI ➤ Host Resources and Applications by Shell ➤ IIS Applications by NTCMD ➤ MQ by Shell ➤ MS Cluster by NTCMD To obtain a list of other jobs that include this feature, contact HP Software Support.
Useful Links	➤ "Viewing Permissions While Running Jobs" on page 95 ➤ "Required Permissions Pane" on page 268 ➤ "Permission Editor Dialog Box" on page 271

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
📋▾	Export a permission object in Excel, PDF, RFT, CSV, or XML format. For details, see "Browse Mode" in *Model Management*.
Objects and Parameters	The commands that appear in the relevant Jython scripts.
Operation	The action that is being run.
Permission	The name of the protocol as defined for the job.
Usage Description	A description of how the protocol is used.

Discovery Scheduler Dialog Box

Description	Enables you to define a schedule for a specific job, for example, every day DDM starts running an IP ping sweep on class C networks at 6:00 AM. **To access:** ➤ Right-click a job and choose **Edit scheduling**. ➤ Click the **Edit Scheduler** button in the Discovery Scheduler pane of the Properties tab in the Run Discovery window.
Important Information	The Discovery Scheduler defines the frequency of the discovery (daily, monthly) whereas the time template defines when the job should run (during the day, at night, at weekends only). You can run the same schedule with different time templates. For example, you can define a schedule that runs every day and you can define a time template that runs at night from 01:00 AM to 05:00 AM. A job defined in this way runs every day from 01:00 AM to 05:00 AM. You can define a second time template to run at a different time, and you can use this time template too with the same schedule. For details on creating a time template, see "Edit Time Template Dialog Box" on page 151.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
	Click to validate the Cron expression you entered.
<**Frequency**>	➤ **Once.** Define the task to run only once. ➤ **Interval.** Defines the interval between successive runs. ➤ **Daily.** Run a task on a daily basis. ➤ **Weekly.** Run a task on a weekly basis. ➤ **Monthly.** Run a task on a monthly basis. ➤ **Cron.** Enter a Cron expression in the correct format.

GUI Element (A–Z)	Description
Days of month	(Displayed when you select **Monthly**.) Click the button to choose the days of the month on which the action must run. The Select Days dialog box opens. Choose the required days by selecting the check boxes. You can select multiple days. ➤ **Select all**. Select all the days. ➤ **Unselect all**. Clear all the selected days.
Days of the week	(Displayed when you select **Weekly**.) Select the day or days on which the action should run.
End by	Select the date and time when the action should stop running by selecting the **End by** check box, opening the calendar, selecting the date and time, and clicking **OK**. **Note**: This step is optional. If you do not need to specify an ending date, leave the **End by** check box cleared.
Invocation Hour	(Displayed when you select **Daily**, **Weekly**, or **Monthly**.) Select the time to activate the action. Click the button to open the Select Hours dialog box. Choose the required time by selecting the check boxes. You can select multiple times. ➤ **Select all**. Select all the times. ➤ **Unselect all**. Clear all the selected times. **Note**: You can also enter the time manually in the **Invocation hour** box. Separate times by a comma and enter AM or PM after the hour. The manually entered action times are not restricted to the hour and half hour only: you can assign any hour and minute combination. Use the following format: HH:MM AM, for example, 8:15 AM, 11:59 PM.
Invocation Time	(Displayed when you select **Once**.) Choose the date and time the action should begin running by opening the calendar and choosing a date and time, or accept the default.
Months of the year	(Displayed when you select **Monthly**.) Select the month or months in which the action must run.
Repeat every	(Displayed when you select **Interval**.) Type a value for the interval between successive runs and choose the required unit of time (minutes, hours, or days).

GUI Element (A–Z)	Description
Start at	Choose the date and time when the action must begin running by selecting the **Start at** check box, opening the calendar, selecting the date and time, and clicking **OK**.
Time Zone	Select the time zone according to which the Probe must schedule jobs. The default is <<**Discovery Probe Time Zone**>>: the Probe uses its own system-defined time zone. This enables scheduling to take place at different times in different geographical locations. For all Probes to start working at the same time, select a specific time zone. (This assumes that the Probes' system date and time and time zone are correctly configured.)

Edit Probe Limitation for TQL Output Dialog Box

Description	Enables you to change the Probes on which a trigger TQL is running. For details on selecting the Probes, see "Selecting Probes" on page 203. **To access:** Select a job and click the following button: **Run Discovery** > **Properties tab** > **Trigger TQLs pane** > **Probe Limit** box.

Edit Time Template Dialog Box

Description	Enables you to define a time template to use when scheduling jobs. **To access:** Click the **Add** button in the Time Templates dialog box.
Important Information	The name of the time template must be unique.
Useful Links	"Discovery Scheduler Dialog Box" on page 149

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Every day between	Define a daily schedule when a job must run. You can also type in times. You can assign any hour and minute combination.
Time Template	Enter a unique name.
Week Time	Define a weekly schedule when a job must run. Click to select a time. To select adjacent cells, click and drag the pointer over the table. To clear a time, click a cell a second time.

Find Jobs Dialog Box

Description	Enables you to search for jobs answering to specific criteria. The results of the search are displayed in the Selected Items pane in the Details tab. **To access:** Click the **Search for Discovery Jobs** button in the Discovery Modules pane.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Direction	Searches forwards or backwards through the modules.
Find All	All jobs meeting the search criteria are highlighted.

GUI Element (A–Z)	Description
Find Discovery job by	Choose between: ➤ **Name.** Enter the name, or part of it, of the job. ➤ **Input type.** CIs that triggered the job. Click the button to open the Choose Configuration Item Type dialog box. Locate the CI type that you are searching for. ➤ **Output type.** CIs that are discovered as a result of the activated job.
Find Next	The next job meeting the search criteria is highlighted.

Infrastructure Wizard

Description	Enables you to run discovery on the networks in your system. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery** > **Basic Mode**. Select **Infrastructure Wizard** from the list in the left pane. Click **Configure and Run**.
Wizard Map	The Infrastructure Discovery wizard contains: **Infrastructure Wizard** > Define IP Ranges > Define Credentials > Preferences > Schedule Discovery > Summary

Define IP Ranges

Description	Enables you to set the network range for discovery for each Probe. The results are retrieved from the addresses in the range you define. You can also define IP addresses that must be excluded from a range.
Important Information	Any changes made here affect the global configuration. General information about the wizard is available in "Infrastructure Wizard" on page 153.
Wizard Map	The Infrastructure Discovery wizard contains: Infrastructure Wizard > **Define IP Ranges** > Define Credentials > Preferences > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	For details, see "Add/Edit IP Range Dialog Box" on page 190.
✖	Select a range and click the button to remove the range from the list.
✎	Select a range and click the button to edit an existing range.
Address Ranges	➤ **Range.** For details on the rules for defining ranges, see "Range" on page 192. ➤ **Excluded.** You can exclude part of a range. Select the range and click the **Add** button. In the dialog box, click the Advanced button. For details, see "Exclude Ranges" on page 191.
Discovery Probes	Enables you to view details on the Probe, including range information. You can also add ranges to, or exclude ranges from, the Probe. For details on defining a Probe, see "Domains and Probes Pane" on page 200.

Define Credentials

Description	Enables you to add, remove and edit a credentials set for protocols.
Important Information	➤ You configure a credentials set depending on what must be discovered and which protocols are supported on your site's network. ➤ For a list of protocols, see "Domain Credential References" on page 204. ➤ General information about the wizard is available in "Infrastructure Wizard" on page 153.
Wizard Map	The Infrastructure Discovery wizard contains: Infrastructure Wizard > Define IP Ranges > **Define Credentials** > Preferences > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	Add new connection details for selected protocol type.
✖	Remove a protocol.
✎	Edit a protocol. Click to open the Protocol Parameters dialog box.
⇧ ⇩	Click a button to move a protocol up or down to set the order in which credential sets are attempted. DDM executes all the protocols in the list with the first protocol taking priority.
Protocol	Click to view details on the protocol, including user credentials.

Preferences

Description	Enables you to choose the configuration options to be used during discovery that are activated by the Infrastructure Discovery wizard.
Important Information	General information about the wizard is available in "Infrastructure Wizard" on page 153.
Wizard Map	The Infrastructure Discovery wizard contains: Infrastructure Wizard > Define IP Ranges > Define Credentials > **Preferences** > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
DNS Nameservers	Discovers DNS nameserver machines and the IPs they hold names for. Choose to activate only if zone transfer can be performed from the Probe machine to the nameserver machines, that is, if the appropriate permissions exist on the DNS nameserver machines. **Network implications.** DDM tries to connect to DNS nameserver servers.
Failover Cluster Discovery	Discovers failover clusters including HP Service Guard, Microsoft Cluster Service and Veritas Cluster.

GUI Element (A-Z)	Description
Host Information	Select the host resources to be discovered. These resources can be either physically or logically part of a host. After DDM connects to a host, it discovers the following resources: ➤ For SNMP agents, the relevant Management Information Base (MIBs). ➤ For WMI agents, the relevant Windows Management Instrumentation Query Language (WQL) queries. DDM can also execute shell commands on a machine. **Network implications.** The Software and Services network resources, because of the large quantities of data that they transmit, may cause very high network traffic. For this reason, the default is not to discover them.
Host TCP Connections	Discover TCP communication channels to map dependency relationships between hosts. This discovery requires that at least one protocol has a defined set of credentials. For details, see the previous Define Credentials step. **Network implications.** DDM executes shell commands on a machine to find open ports.

GUI Element (A-Z)	Description
IP Ping Strategy	Choose the strategy for discovering IPs in your environment. This discovery requires that the SNMP protocol be configured in the previous Define Credentials step. ➤ **Send ping request to every address in defined IP range.** Select this option when you know that most of the IP addresses will respond, the network range is small, and most of the IPs in the range are of interest to you (that is, they are part of your network). ➤ **Send ping request only to discoverable IPs in a network.** Select this option when you know that not all IP addresses will respond and the network range is large. In this case, DDM first discovers a network, then sends a ping request to all discovered IPs in that network. **Versions and Limitations.** Verify that you have the correct credentials set for all the machines between the Probe and one of the network's switches.
Network Topology	Activate to discover the connections, on a discovered switch (for example, a host), between a host and its physical port as well as between a host and its logical layout (VLANs, ELANs). This discovery requires that at least one protocol has a defined set of credentials. For details, see the previous Define Credentials step.

GUI Element (A-Z)	Description
Port Scanning	The TCP ports appearing in the **Choose TCP ports for port scanning** list are scanned to discover open server ports. The ports are scanned on every discovered host. You can add new ports to be scanned, and you can remove existing ports from the list. **To choose a port that does not appear in the list:** 1 Click the **Add port** button to open the Add New Port dialog box. 2 Click the **Add port** button and enter the port name and number. 3 Click **OK**. **Network implications.** Note that the scanning process may affect performance on the network. Furthermore, you may need to inform machine owners that DDM will be trying to connect to their machines.
Software Element	Select to discover software elements running on the discovered hosts. As part of software element discovery, the processes and ports that are related to the software element are also discovered. The Software Library dialog box opens. For details, see “Software Library Dialog Box” on page 278. **Network implications.** A search pattern that is too general causes a toll on performance. For example, do not enter a process name that consists of an asterisk (*) only, as such a filter would try and retrieve all the processes running on all machines.

Schedule Discovery

Description	Enables you to define a schedule for a specific job.
Important Information	For details on scheduling DDM, see "Discovery Scheduler Dialog Box" on page 149. General information about the wizard is available in "Infrastructure Wizard" on page 153.
Wizard Map	The Infrastructure Discovery wizard contains: Infrastructure Wizard > Define IP Ranges > Define Credentials > Preferences > **Schedule Discovery** > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✎	You define a time template in the Discovery Scheduler pane of the Properties tab. For details, see "Discovery Scheduler Pane" on page 171.
Allow Discovery to run at	Choose the time at which the job should run.
Repeat Every	Select how often the job should run.

Summary

Description	Enables you to review the definitions before running discovery.
Important Information	Click **Run** to begin DDM. General information about the wizard is available in "Infrastructure Wizard" on page 153.
Wizard Map	The Infrastructure Discovery wizard contains: Infrastructure Wizard > Define IP Ranges > Define Credentials > Preferences > Schedule Discovery > **Summary**

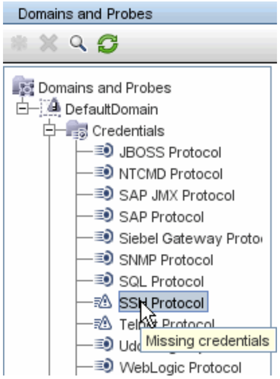
J2EE Wizard

Description	Enables you to run discovery on J2EE applications. **To access: Admin** > **Universal CMDB** > **Discovery** > **Run Discovery** > **Basic Mode**. Select the J2EE wizard from the list in the left pane. Click **Configure and Run**.
Important Information	For more information, hold the pointer over a question mark icon.
Wizard Map	The J2EE Discovery wizard contains: **J2EE Wizard** > Define Credentials > J2EE Port Scanning > WebLogic > WebSphere > JBoss > Oracle Application Server > Schedule Discovery > Summary

Define Credentials

Description	Enables you to configure connection data for each protocol.
Important Information	➤ You configure protocols depending on what must be discovered and which protocols are supported on your site's network. ➤ For a list of protocols, see "Domain Credential References" on page 204. ➤ General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > **Define Credentials** > J2EE Port Scanning > WebLogic > WebSphere > JBoss > Oracle Application Server > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	Add new connection details for selected protocol type.
✖	Remove a protocol.
✎	Edit a protocol. Click to open the Protocol Parameters dialog box.
⇧ ⇩	Move a protocol up or down. DDM executes all the protocols in the list with the first protocol taking priority.
Protocol	Click to view details on the protocol, including user credentials. **Note**: A missing credential is represented by an icon ⚠, as shown in the following image:  Domains and Probes Domains and Probes DefaultDomain Credentials JBOSS Protocol NTCMD Protocol SAP JMX Protocol SAP Protocol Siebel Gateway Proto SNMP Protocol SQL Protocol SSH Protocol Tel Protocol Ud Missing credentials WebLogic Protocol

J2EE Port Scanning

Description	Enables you to choose the port number and port type through which to connect to the J2EE application.
Important Information	General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > **J2EE Port Scanning** > WebLogic > WebSphere > JBoss > Oracle Application Server > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to add a port to the port list. The Add New Port dialog box opens. Select the ports and click OK. To edit existing system ports, in the Add New Port dialog box, click **Edit Known System Ports**. The Edit Known System Ports dialog box opens. Select the port and click the **Edit** button. In the dialog box that opens, make changes to the entries and click **OK**. To add a port to the list, in the Edit Known System Ports dialog box, click the **Add** button. Enter details of the port name, number and type and click **OK**.
✖	Select a port and click the button to remove the port from the list.

WebLogic

Description	Enables you to select the JAR files for specific WebLogic versions.
Important Information	DDM supports the following WebLogic versions: 6.x, 7.x, 8.x, 9.x, and 10.x. 1 To discover WebLogic, obtain the following drivers: ➤ **weblogic.jar** (versions 6.x, 7.x, and 8.x only) ➤ **wlcipher.jar** (if WebLogic is running on SSL, for all versions) ➤ **license.bea** (if WebLogic is running on SSL but only for versions 6.x, 7.x, and 8.x) ➤ **client trust store JKS file** (for example, DemoTrust.jks, but only if WebLogic is running on SSL) ➤ **wlclient.jar** (versions 9.x and 10.x only) ➤ **wljmxclient.jar** (versions 9.x and 10.x only) 2 Place the driver under the correct version folder in the following location: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryResources\j2ee\weblogic\<version_folder>**. For example, **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryResources\j2ee\weblogic\6.x**. 3 Restart the Probe console before running the DDM jobs. 4 On the WebLogic page in the J2EE wizard, select the check box for the versions to be discovered. Click **Import file...** to open a browse window. Browse to the appropriate WebLogic JAR file, as listed below. General information about the wizard is available in “J2EE Wizard” on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > **WebLogic** > WebSphere > JBoss > Oracle Application Server > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Activate using default JAR files (version 8.x only)	Select to enable discovery without specifying version-specific JAR files. This is less recommended and works on some environments only.
WebLogic version 6.x	➤ weblogic.jar ➤ For an SSL based discovery, select wlcipher.jar, license.bea, and the JKS file (for example, DemoTrust.jks)
WebLogic version 7.x	➤ weblogic.jar ➤ For an SSL based discovery, select wlcipher.jar, license.bea, and the client trust store JKS file (for example, DemoTrust.jks)
WebLogic version 8.x	➤ weblogic.jar ➤ For an SSL based discovery, select wlcipher.jar, license.bea, and the client trust store JKS file (for example, DemoTrust.jks)
WebLogic version 9.x	➤ wlclient.jar ➤ wljmxclient.jar ➤ For an SSL based discovery, select wlcipher.jar and the client trust store JKS file (for example, DemoTrust.jks)
WebLogic version 10.x	➤ wlclient.jar ➤ wljmxclient.jar ➤ For an SSL based discovery, select wlcipher.jar and the client trust store JKS file (for example, DemoTrust.jks)

WebSphere

Description	Enables you to select the JAR files for specific WebSphere versions.
Important Information	DDM supports the following WebSphere versions: 5.x, 6.0, and 6.1. ➤ To discover WebSphere, obtain the following certificates: ➤ **client key store JKS file** (DummyClientKeyFile.jks if WebSphere is running on SSL and the file is required) ➤ **client trust JKS file** (DummyClientTrustFile.jks if WebSphere is running on SSL) Out-of-the-box drivers are located on the Probe machine at the following location: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryResources\j2ee\websphere** ➤ Restart the Probe console before running the DDM jobs. General information about the wizard is available in “J2EE Wizard” on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > WebLogic > **WebSphere** > JBoss > Oracle Application Server > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Activate using default JAR files (versions 5.x, 6.x only)	Select to enable discovery without specifying version-specific JAR files. This is less recommended and works on some environments only.
WebSphere	Select the check box for the versions to be discovered. Click **Import file...** to open a browse window. Browse to the appropriate WebSphere JAR file, as follows: ➤ admin.jar ➤ com.ibm.mq.pcf.jar ➤ ffdc.jar ➤ iwsorb.jar ➤ j2ee.jar ➤ jflt.jar ➤ jmxc.jar ➤ jmxx.jar ➤ log.jar ➤ mail.jar ➤ ras.jar ➤ sas.jar ➤ security.jar ➤ soap.jar ➤ utils.jar ➤ wasjmx.jar ➤ websphere_arm_util.jar ➤ wlmclient.jar ➤ wsexception.jar ➤ wssec.jar

JBoss

Description	Enables you to select the JAR files for specific JBoss versions.
Important Information	DDM supports the following JBoss versions: 3.x, 4.x. General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > WebLogic > WebSphere > **JBoss** > Oracle Application Server > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Activate using default JAR files (version 3.x, 4.x only)	Select to enable discovery without specifying version-specific JAR files. This is less recommended and works on some environments only.
JBoss version 3.x and 4.x	Select the check box for the versions to be discovered. Click **Import file...** to open a browse window. Browse to the **jbossall-client.jar** JBoss JAR file.

Oracle Application Server

Description	Enables you to discover Oracle application servers.
Important Information	General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > WebLogic > WebSphere > JBoss > **Oracle Application Server** > Schedule Discovery > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
Discover Oracle Application Server (version 10g)	Select to run DDM for the Oracle Application Server, version 10g.

Schedule Discovery

Description	Enables you to define a schedule for a specific job.
Important Information	General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > WebLogic > WebSphere > JBoss > Oracle Application Server > **Schedule Discovery** > Summary

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✎	You define a time template in the Discovery Scheduler pane of the Properties tab. For details, see "Discovery Scheduler Pane" on page 171.
Allow Discovery to run at	Choose the time at which the job should run.
Repeat Every	Select how often the job should run.

Summary

Description	Enables you to review the definitions before running discovery.
Important Information	To make changes to the run, click the **Back** button. General information about the wizard is available in "J2EE Wizard" on page 161.
Wizard Map	The J2EE Discovery wizard contains: J2EE Wizard > Define Credentials > J2EE Port Scanning > WebLogic > WebSphere > JBoss > Oracle Application Server > Schedule Discovery > **Summary**

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Run	Click to run DDM.

Properties Tab

Description	Enables you to view and administer the properties of modules and jobs. **To access:** Click the **Properties** tab in Run Discovery.
Important Information	Depending which level you select in the Discovery Modules pane, different information is displayed in the Properties tab. If you select: ➤ **The Discovery Modules root**, all active jobs are displayed with scheduling information. Click any of the columns to sort the list by that column. Right-click a job to edit its scheduling. For details, see "Discovery Scheduler Dialog Box" on page 149. ➤ **A Discovery module**, the Description and Module Jobs panes are displayed. To edit a description, make changes in the Description pane and click **OK**. See also "Module Jobs Pane" on page 173. ➤ **A job**, the Parameters, Trigger TQLs, Global Configuration Files, and Discovery Scheduler panes are displayed. For details, see "Parameters Pane" on page 175, "Trigger TQLs Pane" on page 176, "Global Configuration Files Pane" on page 267, and "Discovery Scheduler Pane" on page 171.

Discovery Scheduler Pane

Description	Enables you to view information about the schedule set up for this job. **To access:** Select a job in the Discovery Modules pane in the Run Discovery window.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✎	Click to add times to the Enable Discovery to run at list. The Time Templates dialog box opens. To add a time template to the list, in the Time Templates dialog box, click the **Add** button to open the Edit Time Template dialog box. For details, see “Edit Time Template Dialog Box” on page 151.
Edit Scheduler	Click to open the Discovery Scheduler. For details, see “Discovery Scheduler Dialog Box” on page 149.
Invoke on New Trigger CIs Immediately	A check mark signifies that the job runs as soon as the Trigger CI reaches the Probe. If this column does not contain a check mark, the job runs according to the schedule defined in the Schedule Manager.
Time Template	Choose a template that includes the days and times when the job should run.

Global Configuration Files Pane

For details, see “Global Configuration Files Pane” on page 267.

Module Jobs Pane

Description	Enables you to view the active jobs for a specific module. **To access:** Select a module in the Discovery Modules pane in the Run Discovery window.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

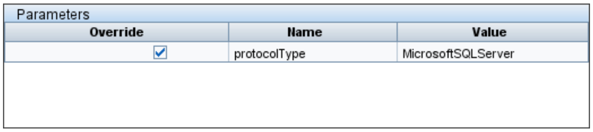
GUI Element (A–Z)	Description
➕	**Add Discovery Job to Module.** Opens the Choose Discovery Jobs dialog box where you can select jobs from more than one zip file. (Use the SHIFT or CTRL key to select several jobs.)
❌	**Remove Selected Discovery Job from Module.** Select the job and click the button. (No message is displayed. To restore the job, click the **Cancel** button.)
🗺	**Show results as a map**. You can choose to view a map of the CIs and links that are discovered by the pattern, instead of a list. Click the button to open the Discovered Classes Map window. The selected pattern is shown together with its CIs and relationships. Hold the cursor over a CIT to read a description in a tooltip.
<Column title>	Click a column title to change the order of the CITs from ascending to descending order, or vice versa.
<List of jobs>	All jobs included in the module. (Displayed when a specific module is selected in the Discovery Modules pane.)

GUI Element (A–Z)	Description
<right-click menu>	Right-click a row to open the Discovery Scheduler for the selected job. For details, see “Discovery Scheduler Dialog Box” on page 149. Right-click a column title to customize the table. Choose from the following options: ➤ **Hide Column.** Select to hide a specific column. ➤ **Show All Columns.** Displayed when a column is hidden. ➤ **Customize.** Select to display or hide columns and to change the order of the columns in the table. Opens the Columns dialog box. ➤ **Auto-resize Column.** Select to change a column width to fit the contents. For details, see “Select Columns Dialog Box” in *Reference Information*.
Invoke Immediately	➤ A check mark signifies that the Discovery job runs as soon as the triggered CI reaches the Probe. In this case, the Invoke on new triggered CIs immediately check box is selected in the Properties tab. ➤ If this column does not contain a check mark, the job runs according to the schedule defined in the Schedule Manager.
Job Name	The name of the job and the package in which the job is included. (Displayed when a job is selected in the Discovery Modules pane.)
Schedule Information	The scheduling information of the job as defined in the Discovery Scheduler.
Trigger TQLs	The name of the TQL that activated the job.

Parameters Pane

Description	Enables you to override pattern behavior. To view a description, hold the pointer over the parameter. **To access:** Select a job in the Discovery Modules pane in the Run Discovery window.
Important Information	You can override a default pattern parameter for a specific job, without affecting the default value.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

<table>
<tr><th>GUI Element (A–Z)</th><th>Description</th></tr>
<tr><td><b>Name</b></td><td>The name given to the pattern.</td></tr>
<tr><td><b>Override</b></td><td>Select to override the parameter value in the pattern.
When this check box is selected, you can override the default value. For example, to change the protocolType parameter, select the <b>Override</b> check box and change MicrosoftSQLServer to the new value. Click <b>OK</b> in the Properties tab to save the change:

Parameters
<table>
<tr><th>Override</th><th>Name</th><th>Value</th></tr>
<tr><td>☑</td><td>protocolType</td><td>MicrosoftSQLServer</td></tr>
</table>
For details on editing parameters in the Discovery Pattern Parameters pane, see “Discovery Pattern Parameters Pane” on page 266.</td></tr>
<tr><td><b>Value</b></td><td>The value defined in the pattern.</td></tr>
</table>

Trigger TQLs Pane

Description	Enables you to define one or more TQL queries to be used as triggers to activate the selected job. **To access:** ➤ Select a job in the Discovery Modules pane in the Run Discovery window. ➤ Create a job by right-clicking a module in the Discovery Modules pane, and choose **Create New Job**.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	**Add TQL.** You can add one or more non-default TQL queries to be used as triggers to activate the selected job. Click to open the **Choose Discovery TQL** dialog box.
❌	**Remove TQL.** Select the TQL and click the button. (No message is displayed. To restore the TQL, click the **Cancel** button.) **Note:** If a TQL query is removed for an active job, DDM no longer receives new CIs coming from that TQL query. Existing Trigger CIs that originally came from the TQL query are not removed.
...	Click to add or remove Probes for a specific TQL. For details, see "Edit Probe Limitation for TQL Output Dialog Box" on page 151.
✎	Click to open the Trigger TQL Editor. For details, see "Trigger TQL Editor Window" on page 179.
🖥	Click to open the Query Manager. For details, see "Query Manager" in *Model Management*.
Probe Limit	The Probes being used for the discovery process. To add or remove Probes, click the button.
TQL Name	The name of the trigger TQL query that activates the job.

Related CIs Window

Description	Enables you to view, in map format, the CIs that are related to a selected CI. **To access:** In the **Discovered CIs** dialog box, right-click a CIT and select **Get Related CIs**.
Important Information	Related CIs are CIs that are the parent, child, or sibling of an existing CI.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
<right-click menu>	For details, see "IT Universe Manager Context Menu" in *Model Management*.
<menu>	For details, see "Toolbar Options" in *Model Management*.
<Topology Map>	For details, see "Topology Map Overview" in *Model Management*.

Show Results for Triggered CI Dialog Box

Description	Enables you to view the results of running an ad-hoc request to the Probe. DDM acquires the results by running the job on a selected trigger CI. In the case of an error, a message is displayed. **To access:** Run Discovery, select a module or job, select the Details tab. In the Discovery Status pane, drill down to a CI, right-click it, and choose **Show Results for Triggered CI**.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
	Select a CIT and click to display additional information in the Show Results for Triggered CI dialog box. For details, see “Show Results for Triggered CI Dialog Box” on page 177.
🔍	Click to open a topology map showing a result map for the Triggered CI. Right-click a CIT to view its properties.

Source CIs Dialog Box

The Source CIs dialog box includes the same components as the **Discovered CIs** dialog box. For details, see “Discovered CIs Dialog Box” on page 144.

Time Templates Dialog Box

Description	Enables you to define a daily or weekly schedule to run selected jobs. **To access: Run Discovery** > **Properties tab** > **Discovery Scheduler pane** > **Time Template icon**.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
+	Click to add a time template. Opens the Edit Time Template dialog box.
✗	Select a time template and click to delete.
✎	Select a time template and click to edit it. Opens the Edit Time Template dialog box.

Triggered CIs Window

Description	Enables you to view all CI instances found for a selected TQL node. **To access:** ➤ **Run Discovery** > **Dependency Map** tab. Right-click a CIT and select **Show Triggered CIs**. ➤ In the Discovery Status pane, click the **Show additional data** button.
Important Information	The Triggered CIs window includes the same information as the Element Instances window. For details, see "Element Instances Dialog Box" in *Model Management*.

Trigger TQL Editor Window

Description	Enables you to edit a TQL that has been defined to trigger jobs. **To access: Run Discovery** > **Properties tab** > **Trigger TQLs pane** > select a TQL and click the **Open the TQL Editor** button.
Important Information	A Trigger TQL associated with a job is a subset of the Input TQL, and defines which specific CIs should be the Trigger CIs for a job. That is, if an Input TQL queries for IPs running SNMP, a Trigger TQL queries for IPs running SNMP in the range 195.0.0.0-195.0.0.10.
Useful Links	➤ "Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs" on page 58 ➤ "Input TQL Editor Window" on page 251

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<Panes>	➤ CI Type Selector Pane ➤ Editing Pane ➤ Information Pane
TQL Name	The name of the trigger TQL query that activates the job.

CI Type Selector Pane

Description	Displays a hierarchical tree structure of the CI Types found in the CMDB. For more details, see "CI Type Manager User Interface" in *Model Management*. **Note**: The number of instances of each CIT in the CMDB is displayed to the right of each CIT. To create or modify a TQL query, click and drag nodes to the Editing pane and define the relationship between them. Your changes are saved to the CMDB. For details, see "Add Nodes and Relationships to a TQL Query" in *Model Management*.
Included in Tasks	➤ "Define a TQL Query" in *Model Management* ➤ "Create a Pattern View" in *Model Management*

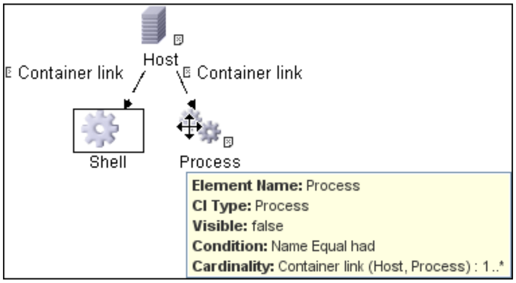
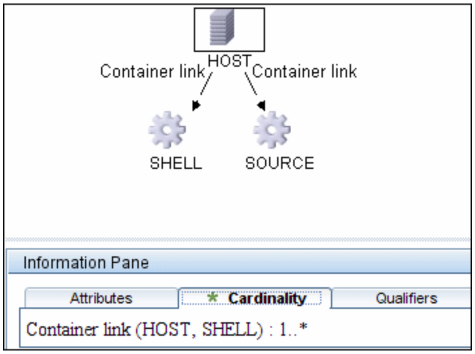
Editing Pane

Description	Enables you to edit the node selected in the Trigger TQLs pane.

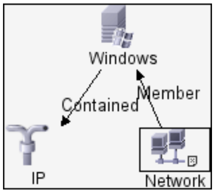
The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<node>	Click to display information about the node in the information pane.
<right-click menu>	For details, see “Context Menu Options” in *Model Management*.
<Toolbar>	For details, see “Toolbar Options” in *Model Management*

Information Pane

Description	Displays the properties, conditions, and cardinality for the selected node and relationship.
Important Information	Hold the pointer over a node to view information:  **Element Name:** Process **CI Type:** Process **Visible:** false **Condition:** Name Equal had **Cardinality:** Container link (Host, Process) : 1..* A small green indicator is displayed next to the tabs that include information:  HOST Container link Container link SHELL SOURCE Information Pane Attributes Cardinality Qualifiers Container link (HOST, SHELL) : 1..*

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
Edit	To view information, select a node or relationship in the Editing pane, select the tab in the Information Pane, and click the **Edit** button. For details on the Node Condition dialog box, see “Node/Relationship Properties Dialog Box” in *Model Management*.
Attributes	Displays the attribute conditions defined for the node or the relationship. For details, see “Attribute Tab” in *Model Management*.
Cardinality	Cardinality defines how many nodes you expect to have at the other end of a relationship. For example, in a relationship between host and IP, if the cardinality is 1:3, the TQL retrieves only those hosts that are connected to between one and three IPs. For details, see “Cardinality Tab” in *Model Management*.
Details	➤ **CI Type.** The CIT of the selected node/relationship. ➤ **Visible.** A tick signifies that the selected node or relationship is visible in the topology map. When the node/relationship is not visible, a box ⊡ is displayed to the right of the selected node/relationship in the Editing pane:  ➤ **Include subtypes.** Display both the selected CI and its descendants in the topology map.
Qualifiers	Displays the qualifier conditions defined for the node or the relationship. For details, see “Qualifier Tab” in *Model Management*.

GUI Element (A-Z)	Description
Selected Identities	Displays the element instances that are used to define what should be included in the TQL results. For details, see “Identity Tab” in *Model Management*.

7

Set Up Discovery Probes

This chapter provides information on setting up Discovery and Dependency Mapping (DDM) Probes.

This chapter includes:

Concepts

Tasks

Reference

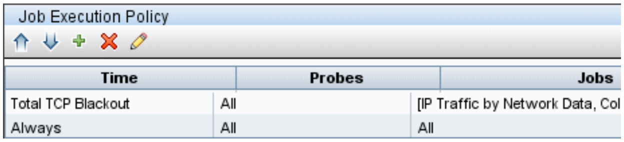
Job Execution Policies

You can define periods of time when a Probe must not run. You can choose to disable specific jobs running on any Probe or all jobs running on a specific Probe. You can also exclude jobs from a job execution policy so that they continue running as usual.

For details on defining a job execution policy, see “Add/Edit Policy Dialog Box” on page 192.

Example of Policy Ordering

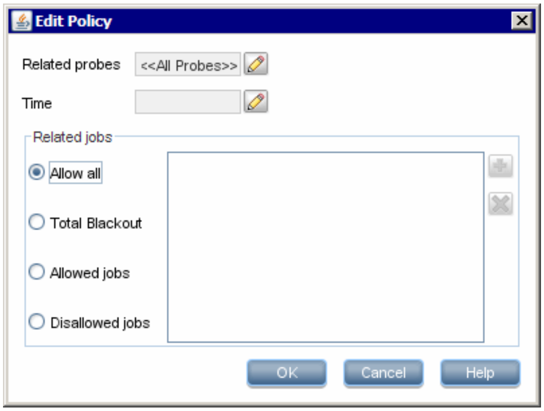
There are two policies, **Total TCP Blackout** and **Always**. **Total TCP Blackout** does not allow any TCP discovery jobs to run. The policies appear in the list as follows:



Job Execution Policy

Time	Probes	Jobs
Total TCP Blackout	All	[IP Traffic by Network Data, Col
Always	All	All

A job (Class C IPs by ICMP) starts running. It checks the policies in the policy list from top to bottom. It starts by checking **Total TCP Blackout**. The job does not appear in this policy, so it continues down the list and checks **Always**. The job does appear here (**Allow All** is selected in the Edit Policy dialog box) so the job runs:



Edit Policy

Related probes <<All Probes>>

Time

Related jobs

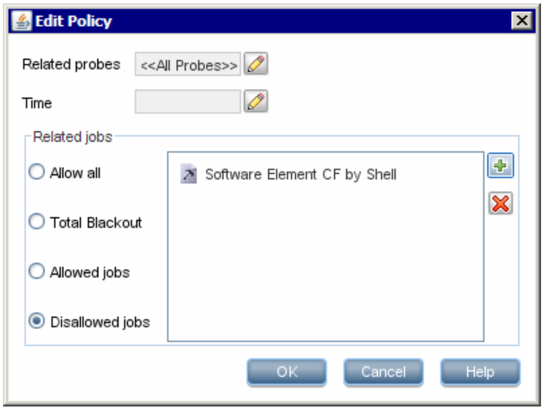
● Allow all

○ Total Blackout

○ Allowed jobs

○ Disallowed jobs

The next job (Software Element CF by Shell) starts running. It checks the policies in the policy list from top to bottom. It starts by checking **Total TCP Blackout**. The job appears in this policy (**Disallowed Jobs** is selected in the Edit Policy dialog box), so the job does not run:



Edit Policy

Related probes <<All Probes>>

Time

Related jobs

○ Allow all

○ Total Blackout

○ Allowed jobs

● Disallowed jobs

Software Element CF by Shell

Important: If a job is not connected to any policy, it does not run. To run these jobs, set the last policy in the list to **Allow All**.

Running Jobs When a Job Execution Policy Is Running

If a policy begins to operate while a Probe is executing a job, the job pauses. When the policy finishes, the job continues to run from where it ceased. For example, say a job contains 10,000 Trigger CIs. The job finishes working on 7,000 of them and then the policy starts to operate. When the job continues (after the policy finishes), it works on the remaining 3,000 Trigger CIs—the job does not start running from the beginning.

Add a Probe

This task describes how to add a Probe to DDM.

This task includes the following steps:

- “Prerequisites” on page 188
- “Add a Domain to DDM” on page 188
- “Add a Probe to the New Domain” on page 188
- “Add Further Probes to the Domain – Optional” on page 188

1 Prerequisites

Verify that the Probe is installed and make a note of its IP address.

2 Add a Domain to DDM

In this step, you create the domain for the new Probe.

a Access the Set Up Discovery Probes window: **Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes**.

b Select **Domains and Probes** and click the **Add Domain or Probe** button to open the Add New Domain dialog box. For details, see “Add New Domain Dialog Box” on page 193.

3 Add a Probe to the New Domain

In this step, you define the Probe and its range.

a Double-click the new domain and select the **Probes** folder.

b Click the **Add Domain or Probe** button to open the Add New Probe dialog box. For details, see “Add New Probe Dialog Box” on page 194.

c Select the new probe and define its IP range. For details, see “Add/Edit IP Range Dialog Box” on page 190.

4 Add Further Probes to the Domain – Optional

You can add more probes to this domain. For details, see the previous steps.

Set Up Discovery Probes User Interface

This section describes:

Add/Edit IP Range Dialog Box

Description	Enables you to set the network range for discovery. The results are retrieved from the addresses in the range you define. You can also define IP addresses that must be excluded from a range. **To access:** Click the **Add IP range** button in the Ranges pane (**Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes** > **Details pane)**.
Important Information	If you define a range that is out of the scope of the network on which the Probe is installed, HP Business Availability Center automatically defines the range of the IP on which the Probe is running. A message informs you that the Probe does not include the Probe range. Answer **Yes** to include the IP address in the range.
Included in Tasks	"Run Discovery – Advanced Mode Workflow" on page 98

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	To exclude an IP range from discovery, click the **Add IP range** button.
✖	To delete the excluded part of an IP range, select the excluded range and click the **Remove IP range** button.
✎	To edit the excluded part of an IP range, click the **Edit IP range** button. For details, see Exclude Ranges.

GUI Element (A–Z)	Description
Exclude Ranges	Click the **Add IP range** or **Edit IP range** button to exclude part of a range. In the Exclude IP Range dialog box, enter the range to exclude. **Note:** ➤ You must enter a range (in the Add/Edit IP Range dialog box) before you can enter the excluded range. ➤ The rules for entering an excluded range are the same as for entering a range. For details, see Range. ➤ Use this feature to divide a network range into several subranges. For example, say a range is 10.0.64.0 – 10.0.64.255. You define three excluded ranges: 10.0.64.45 – 10.0.64.50 10.0.64.65 – 10.0.64.70 10.0.64.89 – 10.0.64.95 Therefore, the ranges to be discovered are: 10.0.64.0 – 10.0.64.44 10.0.64.51 – 10.0.64.64 10.0.64.71 – 10.0.64.88 10.0.64.96 – 10.0.64.255

GUI Element (A–Z)	Description
Range	The rules for defining an IP address range are as follows: ➤ The IP address range must have the following format: start_ip_address – end_ip_address For example: 10.0.64.0 - 10.0.64.57 ➤ The range can include an asterisk (*), representing any number in the range of 0-255. ➤ If you use an asterisk, you do not need to enter a second IP address. For example, you can enter the range pattern 10.0.48.* to cover the range from 10.0.48.0 to 10.0.48.255. ➤ Use an asterisk in the lower bound IP address of the IP range pattern only. (If you use an asterisk in the lower bound IP address and also enter an upper bound IP address, the upper bound IP address is ignored.) ➤ You can use more than one asterisk (*) in an IP address as long as they are used consecutively. The asterisks cannot be situated between two numbers in the IP address, nor can they be substituted for the first digit in the number. For example, you can enter 10.0.*.* but not 10.*.64.*. ➤ Two Probes in the same domain cannot have the same IP address in their range.

Add/Edit Policy Dialog Box

Description	Enables you to add a job execution policy, to disable jobs from running at specific times. **To access: Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes** > **Details pane** > **Job Execution Policy** section. Select an existing policy and click **Edit**, or click the **Add** button.
Useful Links	"Job Execution Policies" on page 185 "Job Execution Policy Pane" on page 198 "Domain Credential References" on page 204

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Related jobs	➤ **Allow all.** Run the job execution policy on all jobs. ➤ **Total blackout.** The policy does not run on any jobs. ➤ **Allowed jobs.** Choose jobs to run even during the configured blackout time. ➤ **Disallowed jobs.** Choose jobs that do not run during the configured blackout time. For allowed and disallowed jobs, click the **Add job** or **Remove job** button to choose specific jobs to be included in, or excluded from, the policy. If you click the **Add job** button, the Choose Discovery Jobs dialog box opens.
Related Probes ✏	The Probes on which to run the policy. Click the button to open the **Edit Related Probes** dialog box to define which Probes are included in the policy.
Time ✏	The date and time during which the policy is active. Click the button to open the **Edit Timetable** dialog box.

Add New Domain Dialog Box

Description	Enables you to add a domain. **To access:** Click the **Add Domain or Probe** button in the Domains and Probes pane.
Important Information	In a version 8.01 or later environment that has been upgraded from version 6.x, to enable data to be modelled similarly as in the previous version, you must define the Probes as belonging to the **External** domain and not to the **Customer** domain.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Description	Enter a description to appear in the Details pane of the Set Up Discovery Probes window.
Domain Type	➤ **Customer.** A private domain used for your site. You can define several domains and each domain can include multiple Probes. Each Probe can include IP ranges but the customer domain itself has no range definition. ➤ **External.** Internet/public domain. A domain that is defined with a range. The external domain can contain only one Probe whose name equals the domain name. However, you can define several external domains in your system.
Name	Enter a unique name for the domain.

Add New Probe Dialog Box

Description	Enables you to add a Probe. **To access:** Click the **Add Domain or Probe** button in the Domains and Probes pane.
Important Information	➤ To add a Probe to an existing domain, select **Probes** in the Domains and Probes pane and click the **Add Domain or Probe** button. ➤ To add a Probe to a new domain, create a domain, then add the Probe to the domain. ➤ Two Probes in the same domain cannot have the same IP address in their range. ➤ When a Probe is activated, it is added automatically and its status changes to connected. For details, see "Launch the Probe" on page 38.

Choose Discovery Jobs Dialog Box

Description	Enables you to choose the jobs that are to be added to, or excluded from, the job execution policy. **To access:** Select **Allowed Jobs** or **Disallowed jobs** in the Edit Policy dialog box and click the button ➕ .

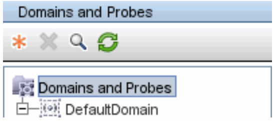
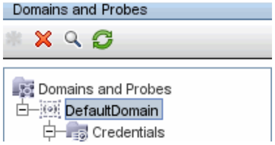
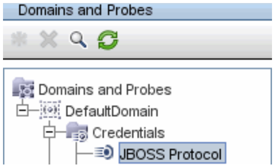
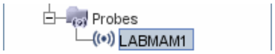
The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<Installed packages>	Locate the job to be included in, or excluded from, the policy. (Use the SHIFT or CTRL key to select several packages.)

Details Pane

Description	Enables you to view the Probes running under all domains and to add an execution policy to jobs (that is, to schedule time periods when jobs should not run). **To access:** Click an object in the Domains and Probes pane.
Important Information	Depending on what you select in the Domains and Probes pane, different information is displayed in the Details tab. For details, see "Displayed Information" on page 196 in the next section.

Displayed Information

If You Select...	The Information Displayed Is...
	**Domains and Probes.** You can view details on all Probes and you can define and edit job execution policies. For details, see “Discovery Probes Pane” on page 197 and “Job Execution Policy Pane” on page 198.
 Domains and Probes Domains and Probes DefaultDomain Credentials	**A specific domain.** You can add a description and view a list of Probes running in that domain. For details, see “Discovery Probes Pane” on page 197 and “Description Pane” on page 196.
 Domains and Probes Domains and Probes DefaultDomain Credentials JBOSS Protocol	**A specific protocol.** You can add protocol parameters and you can view details on the protocol, including user credentials. For details, see “Domain Credential References” on page 204.
	**A specific Probe.** You can view details on the Probe, including range information. You can also add ranges to, or exclude ranges from, the Probe. For details, see “Ranges Pane” on page 199, “Discovery Probes Pane” on page 197, and “Details Pane” on page 197.

Description Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Description	The description that was entered during domain creation.
Domain Type	For details, see Domain Type in “Add New Domain Dialog Box” on page 193.

Details Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Last time Probe accessed	The last time that the Probe was accessed on the server machine.
Probe IPs	The IP of the Probe machine.
Status	➤ **Connected.** The Probe has successfully connected to the server (the Probe connects every few seconds). ➤ **Disconnected.** The Probe is not connected to the server.

Discovery Probes Pane

Description	Enables you to view a list of all Probes connected to the server. **To access:** Click a Probe in the Domains and Probes pane.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
IP	The IP range defined during Probe creation.
Last Access Time	The last time that the Probe requested tasks from the server.
Name	The Probe name as it appears in DDM.
Status	Can be Connected or Disconnected.

Job Execution Policy Pane

Description	Enables you to configure the periods of time when jobs should not run. **To access: Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes**. Select **Domains and Probes**.
Important Information	Jobs that have a listening functionality (that is, they do not perform discovery, for example, they listen to SNMP traps) are not included in a policy.
Useful Links	"Job Execution Policies" on page 185 "Domain Credential References" on page 204

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
⇧ ⇩	Move the policy up or down. DDM executes all the policies in the list with the first policy taking priority. If a job is included in two policies, DDM executes the first policy only for that job.
+	Add a policy.
✖	Remove a policy.
✎	Edit a policy. Click to open the **Edit Policy** dialog box.
Jobs	The jobs that are affected by the policy.
Probes	The Probes that are affected by the policy.
Time	The schedule of the policy.

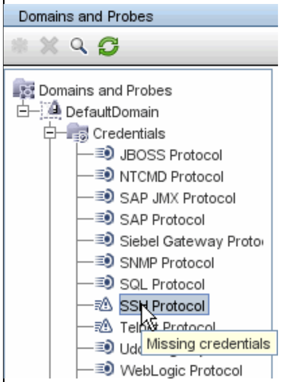
Ranges Pane

Description	Enables you to add and remove ranges that a Probe should work with. **To access:** Click a Probe in the Domains and Probes pane.
Important Information	For details on searching for a specific range, locate the **Find Probe Range by IP** button in "Domains and Probes Pane" on page 200.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to open the **Add IP Range** dialog box.
✖	Click a range and click the button to remove a range from the list.
✎	Click to open the **Edit IP Range** dialog box.
⇦ ▾	Export a permission object in Excel, PDF, RFT, CSV, or XML format. For details, see "Browse Mode" in *Model Management*.
⇨	Click to import ranges from a CSV file. Before using this feature, verify that the imported file is a valid CSV file, and that the ranges in the file do not conflict with existing ranges (that is, there are no duplicate or overriding ranges).
Excluded	Displays the IP addresses that have been excluded from the range that the Probe uses to discover CIs. For details, see "Add/Edit IP Range Dialog Box" on page 190.
Range	The network IP addresses that the Probe uses to discover CIs. For details, see "Add/Edit IP Range Dialog Box" on page 190.

Domains and Probes Pane

Description	Enables you to view, define, or edit a domain, a Probe or a Probe's credentials. **To access: Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes**.
Important Information	A missing credential is represented by an icon ⚠, as shown in the following image:  Domains and Probes Domains and Probes DefaultDomain Credentials JBOSS Protocol NTCMD Protocol SAP JMX Protocol SAP Protocol Siebel Gateway Proto SNMP Protocol SQL Protocol SSH Protocol Tel Protocol Udd Missing credentials WebLogic Protocol
Useful Links	"Job Execution Policies" on page 185

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✱	Adds a domain or Probe, depending on what is selected.
✗	Deletes a domain or Probe, depending on what is selected.
🔍	**Find Probe Range by IP** button. If a Probe has many ranges defined for it, you can locate a specific range: select the Probe and click the button. In the Find Probe Range dialog box, enter the IP address and click the **Find** button. DDM highlights the range in the Ranges pane.
⟳	Updates all domain and Probe information.

Edit Related Probes Dialog Box

Description	Enables you to select specific Probes. **To access:** Click the **Related Probes** button in the Edit Policy dialog box.
Useful Links	"Job Execution Policies" on page 185

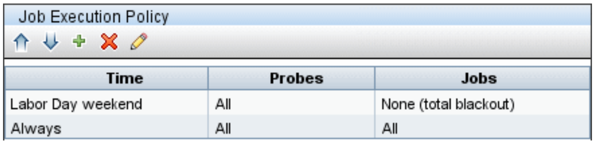
Edit Timetable Dialog Box

Description	Enables you to set the times when a Probe must run a job execution policy. **To access:** Click the **Edit** button in the Edit Policy dialog box.
Useful Links	"Add/Edit Policy Dialog Box" on page 192

The following elements are included (unlabeled GUI elements are shown in angle brackets):

<table>
<tr><th>GUI Element (A–Z)</th><th>Description</th></tr>
<tr><td><b>Description</b></td><td>Add a description of the specific policy. This field is mandatory.

<b>Tip</b>: The text you enter here appears in the Time box in the Job Execution Policy pane, so it is recommended that the description be informative:


Job Execution Policy
<table><tr><th>Time</th><th>Probes</th><th>Jobs</th></tr><tr><td>Labor Day weekend</td><td>All</td><td>None (total blackout)</td></tr><tr><td>Always</td><td>All</td><td>All</td></tr></table></td></tr>
<tr><td><b>Time Definition</b></td><td>Click a cell for a day and time to be included in the policy. To add more than one time unit, drag the pointer over the cells.

<b>Note:</b> To clear a time unit, click the cell a second time.</td></tr>
</table>

Protocol Parameters Dialog Box

Description	Displays the attributes that can be defined for a protocol. **To access: Set Up Discovery Probes** > **Domains and Probes** > **Domain** > **Credentials**, select a protocol and click the **Add** or **Edit** button.
Important Information	For the description of each protocol, see "Domain Credential References" on page 204.

Scope Definition Dialog Box

Description	Enables you to set the range that a protocol must discover. **To access:** Click the **Edit** button in the Protocol Parameters dialog box.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Selected Probes	To select specific Probes whose IP range must be changed, click **Edit**. For details, see "Choose Probe Dialog Box" on page 123.
Selected Ranges	➤ **All.** The protocol runs discovery on all ranges for the domain. ➤ **Selected Range.** For the procedure to select a specific range on which the protocol runs discovery or to define an excluded range, see "Add/Edit IP Range Dialog Box" on page 190.

Set Up Discovery Probes Window

Description	Enables you to define a new domain or to define a new Probe for an existing domain. Also enables you to define the connection data for each protocol. **To access: Admin** > **Universal CMDB** > **Discovery** > **Set Up Discovery Probes.**
Important Information	➤ For details on the Domains and Probes pane, see “Domains and Probes Pane” on page 200. ➤ For details on the Details pane, see “Details Pane” on page 195.
Useful Links	“Domain Credential References” on page 204

Selecting Probes

The Choose Probe to Filter, Edit Probe Limitations for TQL Output, and Edit Related Probes dialog boxes include the following elements (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
>	**Add Selected Probe.** Click to add a Probe to the Selected Probes column.
<	**Remove Selected Probe.** Click to remove a Probe from the Selected Probes column.
All Discovery Probes	➤ Select to add all Probes in the Non-selected Probes list. ➤ Clear to add a specific Probe from the Non-selected Probes list.
Non-selected Probes	Probes that are not included in the policy/filter/limitations.
Selected Probes	Probes that are included in the policy/filter/limitations.

Domain Credential References

This section explains protocol credentials. You can edit credential attributes. For details, see "Protocol Parameters Dialog Box" on page 202.

Note: The following information can change from version to version: Changes in content implementation can cause protocol attributes to be updated.

When a protocol is selected in the Domains and Probes Pane, the following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
➕	Click to add new connection details, to open the **Add Protocol Parameters** dialog box.
❌	Select a protocol and click to remove connection details. Answer **OK** to the message.
✎	Select a protocol and click to edit connection details in the **Protocol Parameters** dialog box.
⇧ ⇩	Select a protocol and click an arrow to move the protocol instance up or down. The order of the policies in the list defines which policy is checked first: a job starts running and checks the policy list from top to bottom. If the job name exists in a policy, the job runs. For details on adding jobs to a protocol, see "Add/Edit Policy Dialog Box" on page 192. For details on job execution policies, see "Example of Policy Ordering" on page 186.

GUI Element (A-Z)	Description
<**Right-click a credential**>	Choose from the following options: ➤ **Edit.** Choose this option to enter protocol parameters, such as user name and password, that enable DDM to connect to an application on a remote machine. ➤ **Edit using previous interface.** Choose this option if: ➤ In a previous version of HP Business Availability Center, you added parameters to this protocol that do not exist in this version. ➤ Values in this version cannot be deleted. For example, in this version you cannot configure SQL Protocol credentials with an empty port number. Select this option to open the previous Edit Protocol Parameter dialog box and delete the port number. ➤ **Check credentials.** In the box that opens, enter the IP address of the remote machine on which the protocol must run. The Probe attempts to connect to this IP and returns an answer as to whether the connection succeeded or not.
<**Right-click a title**>	Choose from the following options: ➤ **Hide Column.** Displayed when a column is shown. ➤ **Show All Columns.** Displayed when a column is hidden. ➤ **Customize.** Select to change the display order of the columns. ➤ **Auto-resize Column.** Select to change the column width to fit the contents.

All protocol credentials include the following parameters:

Parameter	Description
Index	Indicates the order in which protocol instances are used to make a connection attempt. The lower the index, the higher the priority. Default: **9999**. If you do not change the default, this protocol instance is used last.
Network Scope	To change the range that a protocol must discover or to select a Probe, click **Edit**. For details, see "Scope Definition Dialog Box" on page 202. Default: **ALL**.
User Label	Enter a label to help you identify a specific protocol credential, when you use it later. Enter a maximum of 50 characters.

This section includes the following topics:

- "JBoss Protocol" on page 207
- "NNM Protocol" on page 207
- "NTCMD Protocol" on page 208
- "SAP JMX Protocol" on page 208
- "SAP Protocol" on page 209
- "Siebel Gateway Protocol" on page 211
- "SNMP Protocol" on page 211
- "SQL Protocol" on page 213
- "SSH Protocol" on page 214
- "Telnet Protocol" on page 217
- "UDDI Registry Protocol" on page 219
- "VMware Infrastructure Management (VIM) Protocol" on page 219
- "WebLogic Protocol" on page 220

- “WebSphere Protocol” on page 222
- “WMI Protocol” on page 223

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.

NNM Protocol

Parameter	Description
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the NNM server.
NNM Password	The password for the NNM Web service (for example, **Openview**).
NNM User name	The user name of the NNM Web service (for example, **system**).
NNM Webservice Port	The Web service port number of the NNM server (for example, **80**).
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	The user name of the UCMDB Web service (the default is **admin**).

Parameter	Description
UCMDB Webservice Port	The UCMDB Web service port number (the default is **8080**).
UCMDB Webservice Protocol	The protocol for the UCMDB Web service (the default is **http**).

NTCMD Protocol

Parameter	Description
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 name of the domain that includes the host where the Probe is installed.

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.
Connection Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the SAP JMX console.

Parameter	Description
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

<table>
<tr><th>Parameter</th><th colspan="2">Description</th></tr>
<tr><td><b>Timeout</b></td><td colspan="2">Time-out in milliseconds after which the Probe stops trying to connect to the SAP console.</td></tr>
<tr><td rowspan="5"><b>User Name</b></td><td colspan="2">The name of the user needed to log in to the SAP system.
The user should have the following permissions:</td></tr>
<tr><td><b>Authorization Object</b></td><td><b>Authorization</b></td></tr>
<tr><td>S_RFC</td><td>For the <b>S_RFC</b> object, obtain privileges: RFC1, SALX, SBDC, SDIF, SDIFRUNTIME, SDTX, SLST, SRFC, STUB, SUTL, SXMB, SXMI, SYST, SYSU, SEU_COMPONENT.</td></tr>
<tr><td>S_XMI_PROD</td><td>EXTCOMPANY=MERCURY;EXTPRODUCT=DARM;INTERFACE=XAL</td></tr>
<tr><td>S_TABU_DIS</td><td>DICBERCLS=SS; DICBERCLS=SC</td></tr>
<tr><td><b>Password</b></td><td colspan="2">The password of the user needed to log in to the SAP system.</td></tr>
<tr><td><b>SAP Client Number</b></td><td colspan="2">It is recommended to use the default value (800).</td></tr>
</table>

Parameter	Description
SAP System Number	It is recommended to use the default value (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 server where you copied srvrmgr. For details, see "Prerequisites – Copy the driver Tool to the Probe Server" in *Discovery and Dependency Mapping Content Guide*. **Note:** 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.

SNMP Protocol

Parameter	Description
Port	(For SNMP versions v1, v2, and v3) The port number on which the SNMP agent listens.
Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the SNMP agent.
Retry	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.

Parameter	Description
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).
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 (either (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 algorithm is supported.

SQL Protocol

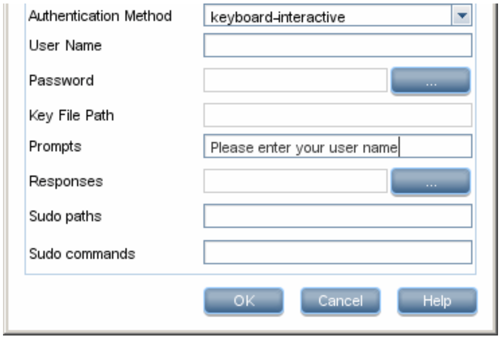
Parameter	**Description**
Database Type	The database type. Select the appropriate type from the box.
Port	The port number on which the database server listens. ➤ If you enter a port number, DDM tries to connect to a SQL database using this port number. ➤ **For an Oracle database**: If there are many 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, DDM 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 "The portNumberToPortName.xml File" on page 230. ➤ The same user name and password is needed to access all Oracle database instances.
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.
Database Name	The database name.

SSH Protocol

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: Discover Windows Running F-Secure" in *Discovery and Dependency Mapping Content Guide*.

Parameter	Description
Port	By default an SSH agent uses port 22. If you are using a different port for SSH, enter that port number.
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.
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. ➤ **keyboard-interactive**. Enter questions and answers. For details, see "Prompts and Responses" on page 216.
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.

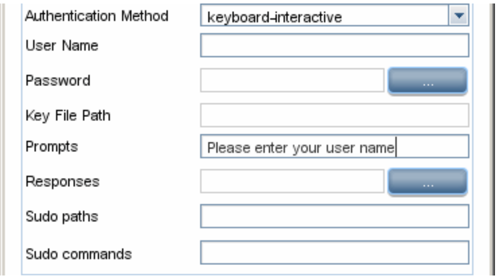
Parameter	Description
Key File Path	(Enabled when the publickey authentication method is selected.) Location of the authentication key. (In certain environments, the full key path is required to connect to an SSH agent.) **Note**: Enter the full path to the key file on the Probe machine.

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, for example:  Authentication Method: keyboard-interactive Prompts: Please enter your user name User Name Password ... Key File Path Responses ... Sudo paths Sudo commands OK Cancel Help or you can enter a key word, for example, **user**. DDM maps this word to the correct prompt.

Parameter	Description
Sudo paths	The full paths to the sudo command. Paths are separated by commas.
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.

Telnet Protocol

Parameter	Description
Port	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.
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 page 216.
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, for example:  Authentication Method: keyboard-interactive User Name Password Key File Path Prompts: Please enter your user name Responses Sudo paths Sudo commands or you can enter a key word, for example, **user**. DDM maps this word to the correct prompt.

Parameter	Description
Sudo paths	The full paths to the sudo command. Paths are separated by commas.
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.

UDDI Registry Protocol

Parameter	Description
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.

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	DDM uses the number defined here when processing one of the Network – VMware jobs: If the port number is left empty, DDM performs a WMI query to extract the port number from the registry. DDM queries **HKLM\SOFTWARE\VMware, Inc.\VMware VirtualCenter** and searches for the **HttpsProxyPort** or **HttpProxyPort** attributes: ➤ If the **HttpsProxyPort** attribute is found, DDM uses its value for the port and sets the prefix to **HTTPS**. ➤ If the **HttpProxyPort** attribute is found, DDM uses its value for the port and sets the prefix to **HTTP**.
Use SSL	**true**: DDM uses a Secure Sockets Layer (SSL) protocol to access VMware Infrastructure, and the prefix is set to **HTTPS**. **false**: DDM uses the http protocol.

Parameter	Description
User Name	The name of the user needed to connect to VMware Infrastructure.
User Password	The password of the user needed to connect to VMware Infrastructure.

WebLogic Protocol

Parameter	Description
Port Number	If you enter a port number, DDM 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, DDM refers to the WebLogic port (defined in portNumberToPortName.xml) already found on this machine (by TCP scanning, using the Network Connection – Active Discovery module). **Note**: You can leave the port number empty on condition that: ➤ All WebLogic ports are added to the portNumberToPortName.xml file. For details, see "The portNumberToPortName.xml File" on page 230. ➤ 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 DDM should connect to the server securely. Enter either **http** or **https**.

Parameter	**Description**
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 Discovery resources folder: <**Discovery Probe installation directory**>\root\lib\collectors\probeManager\discoveryResources\j2ee\weblogic\\<**WebLogic version**>. ➤ Insert the trust store file full path.
Trust Store Password	The SSL trust store password.
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 Discovery resources folder: <**Discovery Probe installation directory**>\root\lib\collectors\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	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.
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 Name	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 Discovery resources folder: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryResources\j2ee\websphere**. ➤ Insert the trust store file full path.
Trust Store Password	The SSL trust store password.

Parameter	Description
Key Store File Name	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 Discovery resources folder: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryResources\j2ee\websphere**. ➤ Insert the keystore file full path.
Key Store Password	The password for the keystore file.

WMI Protocol

Parameter	Description
Timeout	Time-out in milliseconds after which the Probe stops trying to connect to the WMI agent.
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 name of the domain to which the user needed for the connection belongs.

8

Manage Discovery Resources

This chapter provides information on managing Discovery and Dependency Mapping resources such as patterns and configuration files.

This chapter includes:

Concepts

Tasks

Reference

Automatically Deleted System Components

During discovery, the DDM Probe compares CIs found during the previous, successful invocation with those found during the current invocation. A missing component, such as a disk or software, is assumed to have been removed from the system, and its CI is deleted from the Probe's database.

The DDM Probe does not wait for the aging mechanism to perform the calculation but immediately sends a deletion request to the server. For details on aging, see "The Aging Mechanism – Overview" in *Model Management*.

You can define that CI instances are to be deleted for specific jobs. For details, see "Configure the DDM Probe to Automatically Delete CIs – Workflow" on page 231.

By default, the DDM Probe deletes the CI instances of the following CITs:

Job	Pattern	CIT/Link Type
Host Connection By SNMP	SNMP_NET_Dis_Connection	Contained, SNMP
Host Connection By Shell	Host_Connection_By_Shell	Contained, Shell
Host Connection By WMI	WMI_NET_Dis_Connection	Contained, WMI
Host Resources and Applications by SNMP	SNMP_HR_All	Disk, Service, Software, OS User
Host Resources and Applications by WMI	WMI_HR_All	Disk, Service, Software, OS User
Host Resources and Applications by Shell	TTY_HR_All	Disk, Service, Software, OS User, dir

Note:

- The change is defined on the job's pattern.
- If discovery fails and errors occur, no objects are sent for deletion.
- Carefully choose the CIs that are to be candidates for deletion. For example, process CITs are not good candidates because they are often shutting down and starting up again and as a result may be deleted at every invocation.
- You can use this procedure to delete relationships, too. For example, the contained relationship is used between a host and an IP address. A laptop machine is allocated a different IP address very often; by deleting the relationship, you prevent the accumulation of old IP addresses attached to this host.

Example of Automatic Deletion

During the previous invocation, the DDM Probe ran the **Host Resources and Applications by WMI** job and discovered a host with disks a, b, c, and d. During the current invocation, the Probe discovers disks a, b, and c, compares this result with the previous result, and deletes the CI for disk d.

More Information

- You can view deleted CIs in the Probe log and in the Deleted column in the Statistics Results pane. For details, see "Probe Logs" on page 71 and "Statistics Results Pane" on page 142.
- For details on aging, see "The Aging Mechanism – Overview" in *Model Management*.
- An invocation is run according to the Scheduler. For details, see "Discovery Scheduler Dialog Box" on page 149.

Discovering Software Elements

You can discover software (for example, a specific Oracle database) running in your environment.

This section includes the following topics:

Discovery Process

The discovery process runs as follows:

- The Software Element jobs are activated.
- DDM searches for processes on the machines in your environment.
- DDM saves the process data (including open port and command line information) to the Probe database.
- The jobs run on this data in the Probe database, build the new Software Element CIs according to the data in the database, and extract the key attributes from the process data. The jobs send the CIs to the UCMDB server.

Software Element Default Views

Two default views exist that display the mapping of relationships between applications: **Application Components** and **Application Components Dependencies**. To access the views: **Admin** > **Universal CMDB** > **Modeling** > **View Manager** > **Root** > **Application** > **Application Signature**.

You can configure DDM to discover software elements. For details, see "Discover Software Elements – Scenario" on page 232.

Identifying Software Element Processes

You can select key processes to enable DDM to:

- Identify which applications should be discovered.
- Create the appropriate software element CI.

If you do not define any key processes, DDM searches for other processes:

- If none are found, DDM does not create any CIs.
- If at least one process is found, DDM creates a process CI—but only on condition that it also discovers a software element CI (that is, the process must be linked to a software element CI).

For example, say that DDM has discovered signatures for two applications, application **A** and application **B**:

- Application **A** includes processes: wrapper.exe, mainA.exe.
- Application **B** includes processes: wrapper.exe, mainB.exe.

If no key process is defined, and **wrapper.exe** is found, both software elements are created (application **A** and application **B**).

If a key process is defined (in this case, mainA.exe or mainB.exe), DDM discovers this key process. If DDM discovers mainA.exe, it creates a software element CI for Application **A** and if it discovers mainB.exe, it creates a CI for Application **B**.

If DDM also discovers wrapper.exe, DDM creates a process CI and a link from the software element CI to the process CI.

For details on the key field in the Software Identification Rule Editor dialog box, see “Identifying Processes” on page 277.

The portNumberToPortName.xml File

The portNumberToPortName.xml file is used by DDM as a dictionary to create Port 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 Port CI with that name. If the port name does not appear in this file, the Probe uses the port number as the port name.

For details on adding new ports to be discovered, see “Define a New Port” on page 235.

Note:

- The portNumberToPortName.xml file may include a **cpVersion** attribute. This attribute is intended for internal use only by HP Software Support, and indicates in which Content Pack version the port was discovered. When deploying the **portNumberToPortName.xml** file, UCMDB performs a smart merge between the files; entries added by the user are not removed.
- The results of running a Network Connections – Active discovery appear in the Topology Map with the port names instead of the port numbers (the port title is the value of the Port Name attribute, defined in the CIT). For details, see “Add/Edit Attribute Dialog Box” in *Model Management*.

Configure the DDM Probe to Automatically Delete CIs – Workflow

This task explains how to configure a job so that CI instances of specific CITs are automatically deleted. For details on how the DDM Probe deletes CIs, see “Automatically Deleted System Components” on page 226.

This task includes the following steps:

- “Prerequisites” on page 231
- “Select the CIs to be Deleted” on page 231
- “Results” on page 231

1 Prerequisites

Verify that the **Filter unchanged results** check box is selected in the Results Management pane in the Pattern Management tab. For details, see “Results Management Pane” on page 264.

2 Select the CIs to be Deleted

a Select the **Automatically delete removed CIs** check box.

b Click the **Add** button to open the Choose Discovered Class dialog box. For details, see “Choose Discovered Class Dialog Box” on page 241.

c Click the **Save** button at the bottom of the page.

3 Results

To view the deleted CIs, access the Deleted column in the Statistics Results pane. For details, see “Statistics Results Pane” on page 142.

Discover Software Elements – Scenario

This scenario explains how to set up the discovery of Oracle databases so that there is no need to enter a specific set of credentials to discover each database instance. DDM runs an extract command that retrieves the database name attribute.

In this scenario, we assume that the following syntax is used in the Oracle command lines:

```
c:\ora10\bin\oracle.exe UCMDB
```

This task includes the following steps:

- “Prerequisites” on page 232
- “Create a Command Line Rule” on page 233
- “Define the Value of an Attribute” on page 233
- “Activate the Job” on page 234

1 Prerequisites

Display the Software Element Attribute Assignment Rules dialog box:

a Select **Admin** > **Universal CMDB** > **Discovery** > **Run Discovery**. In the **Discovery Modules** pane, select **Host Resources and Application Dependency** > **Software Element CF by Shell**. In the Properties tab, select **Global Configuration Files** > **applicationSignature.xml**. For details, see “Global Configuration Files Pane” on page 267.

b Click the **Edit** button to open the Software Library dialog box. For details, see “Software Library Dialog Box” on page 278.

c Select **Software Element Categories** > **Database** > **Oracle by oracle.exe process**. Click the **Edit** button to open the Software Identification Rule Editor dialog box. For details, see “Software Identification Rule Editor Dialog Box” on page 276.

d Click the **Set Attributes** button to open the Software Element Attributes Assignment Rules dialog box. For details, see “Software Element Attribute Assignment Rules Dialog Box” on page 275.

2 Create a Command Line Rule

The command line rule is text that identifies the process to be discovered, for example, oracle.exe c:\ora10\bin\oracle.exe UCMDB. You can substitute the text entry with a regular expression, so that discovery is more flexible. For example, you can set up a rule that discovers all Oracle databases, whatever their name.

Subsequently, DDM uses the information in the command lines discovered by the regular expression to populate a CI's data_name attribute with the database name.

a To create a Command Line rule that includes a regular expression, in the Software Element Attributes Assignment Rules dialog box, click the **Add** button in the Parsing Rules pane. For details, see "Parse Rule Editor Dialog Box" on page 257.

b In the Parse Rules Editor dialog box, build the rule:

- Enter a unique name in the Rule ID field, for example, **r1**.
- Choose **Command Line** in the Process Attribute field.
- Enter the following regular expression in the Regular Expression field: **.+\s+(\w+)$**:

  This expression searches for any character (**.**), followed by a space or spaces (**+\s+**), followed by a word or words (**(\w+)**) that appear at the end of the line (**$**). You can use the following characters: a-z, A-Z, or 0-9. The following command line fulfils this expression: c:\ora10\bin\oracle.exe UCMDB.

3 Define the Value of an Attribute

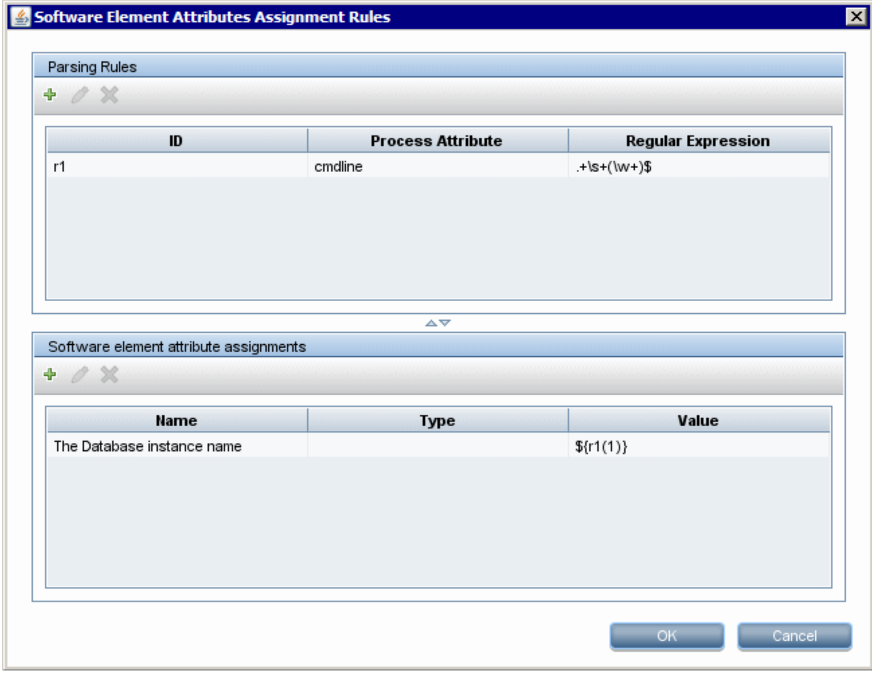
In this step, you define which attribute is used by DDM to discover the Oracle databases, and the value it should take.

a In the Software Element Attributes Assignment Rules dialog box, click the **Add** button in the Software Element Attribute Assignments pane, to select the attribute.

b In the Attribute Editor dialog box:

- Choose the attribute that holds the database name, from the list of Oracle CIT attributes, in this case **The Database instance name**.

➤ Enter a value, using the following syntax: **${<rule ID name>(<group number>)}**, in this case, **${r1(1)}**.



Software Element Attributes Assignment Rules
Parsing Rules
Software element attribute assignments
OK
Cancel

ID	Process Attribute	Regular Expression
r1	cmdline	.+\s+(\w+)$

Name	Type	Value
The Database instance name		${r1(1)}

This dialog box is configured as follows: DDM should search for Oracle databases where the value of the database name attribute (**${r1(1)}**) in the command line is equal to the first group (**(\w+)$**) in the regular expression (for details, see step b on page 233 in “Create a Command Line Rule”).

That is, during discovery, DDM searches through the configuration files for command lines with a word or words at the end of the line. For example, the following command line matches this regular expression: c:\ora10\bin\oracle.exe UCMDB.

4 Activate the Job

For details, see “Manually Activate a Job” on page 67 and “Discovery Modules Pane” on page 144.

Define a New Port

For an explanation of the portNumberToPortName.xml file, see “The portNumberToPortName.xml File” on page 230.

1 In the Manage Discovery Resources window (**Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources**), search for the portNumberToPortName.xml file: click the **Find resource** button and enter **portNumber** in the **Name** box. Click **Find Next**, then click **Close**.

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

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

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

- ➤ **portProtocol.** The network protocol used for discovery (udp or tcp).
- ➤ **portNumber.** The port number to be discovered.
- ➤ **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**. Used to add a version attribute to a port, so that a new entry is not skipped when the Probe creates port CIs from newly-discovered ports.

Resource Files

The following files can be changed to enable DDM in non-default systems. The location of these files is: **Manage Discovery Resources** > **Network** > **Configuration Files**.

This section includes the following topics:

oidToHostClass.xml

The oidToHostClass.xml file contains a list of OID (Discovery and Dependency Mapping) numbers, for all CIs in the system that have an ID. This list is required for mapping CIs to their correct CIT, and for converting the discovered OID number of an operating system or a device into string data.

To access the oidToHostClass.xml file, in Manage Discovery Resources, search for the file by clicking the **Find resource** button and entering **oidto** in the **Name** box. Click **Find Next**, then click **Close**.

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

Note: If an OID is discovered and its details do not appear in the oidToHostClass.xml file, its CIT is registered in the CMDB as host.

The oidToHostClass.xml file includes the following parameters:

- **class.** The converted CIT name of the discovered OID. Under this name, the operating system or device appears in the CMDB and in Business Availability Center.
- **vendor.** The vendor of the operating system or device.
- **os.** A specific operating system, for example, Linux. This parameter is optional.
- **model.** A specific model, for example, JETDIRECT,JD30. This parameter is optional.
- **oid.** The discovered OID.

globalFiltering.xml

This file enables you to filter Probe results for all patterns, so that only results of interest to you are sent to the Business Availability Center server. (You can also filter specific patterns. For details, see “Pattern Management Tab” on page 258.)

To add a global filter:

1. Access the globalFiltering.xml file: in Manage Discovery Resources, open the Network folder and click the Configuration Files folder. Select the file to display the code in the View pane.
2. Locate the <includeFilter> and <excludeFilter> markers:

   ➤ **<includeFilter>.** When a vector marker is added to this filter, all CIs that do not match the filter are removed. If this marker is left empty, all results are sent to the server.

   ➤ **<excludeFilter>.** When a vector marker is added to this filter, all CIs that match the filter are removed. If this marker is left empty, all results are sent to the server.

   The following example shows an ip CI that has address and domain attributes:

```
<vector>
    <object class="ip">
        <attribute name="ip_address" type="String">192\.168\.82\.17.*</attribute>
        <attribute name="ip_domain" type="String">DefaultProbe</attribute>
    </object>
</vector>
```

   If this vector is defined in <includefilter>, all results not matching the filter are removed. The results sent to the server are those where the ip_address matches the regular expression **192\.168\.82\.17.*** and the ip_domain is **DefaultProbe**.

   If this vector is defined in <excludefilter>, all results matching the filter are removed. The results sent to the server are those where the ip_address does not match the regular expression **192\.168\.82\.17.*** and the ip_domain is not **DefaultProbe**.

   The following example shows a network CI that has no attributes. All network results are sent to the server:

```
<vector>
    <object class="network">
    </object>
</vector>
```

Note:

- Attributes in the filter should be of type string only. For details on attribute types, see “Attributes Page” in *Model Management*.
- A result is considered to be a match only if all filter attributes have the same values as those in the CI. (If one of a CI’s attributes is not specified in the filter, all the results for this attribute match the filter.)
- A CI can match more than one filter. The CI is removed or remains according to the filter in which it is included.
- DDM filters first according to the <includeFilter> and then applies the <excludeFilter> on the results of <includeFilter>.

Internal Configuration Files

The following files are for internal use only and should be changed only by users with an advanced knowledge of content writing. The location of these files is: **Manage Discovery Resources** > **AutoDiscovery** > **Configuration Files**.

- **discoveryPolicy.xml.** Includes the schedule when the Probe does not execute tasks. For details, see “Add/Edit Policy Dialog Box” on page 192.
- **jythonGlobalLibs.xml.** A list of default Jython global libraries that DDM loads before running scripts.

Manage Discovery Resources User Interface

This section describes:

Attribute Editor Dialog Box

Description	Enables you to define a rule that discovers a CIT according to an attribute. The attribute is defined according to a regular expression. **To access: Software Element Attributes Editor dialog box** > **Software Element Additional Attributes pane**. Click the **Add** button.
Included in Tasks	"Discover Software Elements – Scenario" on page 232
Useful Links	"Parse Rule Editor Dialog Box" on page 257

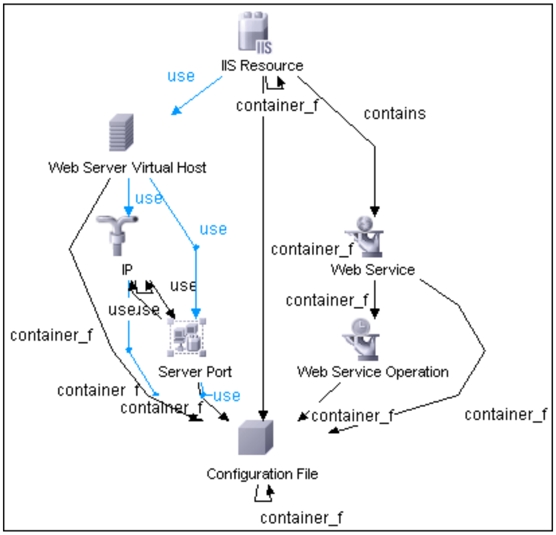
The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Name	Choose from the list of attributes of the CIT selected in the Software Element Editor. This attribute name is replaced by the value found by the regular expression.
Type	The type of operation defined for the attribute, for example, Boolean, string, date, and so on.
Value	The value that replaces the name in the Rule ID field in the Parse Rule Editor dialog box. Use the following syntax for the value: ${<rule ID name>(<group number>)} For example, **${DB_SID(1)}** means that DDM should search for the Rule ID with the name **DB_SID** and retrieve its regular expression. DDM should then retrieve the code for the first group (**1**). For example, in the regular expression **.+\s+(\w+)$**, the first group is **(\w+)$**, that is, a word or words that appear at the end of the line.

Choose Discovered Class Dialog Box

Description	Enables you to choose CITs that are to be discovered by a selected pattern and to limit links so that they are mapped only when they connect specific CITs. **To access:** ➤ **Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources**. In the Discovery Resources pane, select a pattern. In the **Pattern Signature** tab > **Discovered CITs** pane, click the **Add Discovered CIT** button. ➤ **Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources**. In the Discovery Resources pane, select a pattern. In the **Pattern Management** tab > **Automatic Deletion** pane, select the **Enable Automatic deletion of removed CIs** check box and click the **Add** button.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

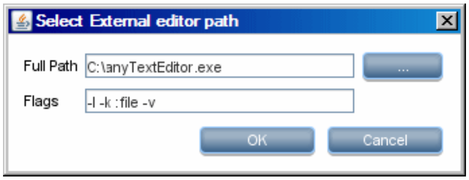
GUI Element (A–Z)	Description
Link	Select a link type from the list and click the button ... in the End 1 and End 2 boxes to open the Choose Configuration Item Type dialog box. Choose the CITs that DDM should map when they are linked by the selected link type. **Note:** DDM automatically recognizes the links between CIs and adds them to the map of discovered CITs. However, during pattern writing, you may need to exclude links between certain CITs. For example, both hosts and IPs and hosts and ports are linked by **use**. You may need to receive results only for hosts and IPs that are connected by the **use** link, and not hosts and ports. The End 1 and End 2 links determine the result received from the pattern, and this result is reflected in the map, as can be seen in the following example:  IIS Resource use container_f contains Web Server Virtual Host use use IP use useuse container_f Server Port container_f container_f use container_f Web Service container_f Web Service Operation container_f container_f Configuration File container_f
Object	Select a CIT to be added to the list of CITs that a pattern is to discover. Save the changes by clicking the **Save** button at the bottom of the Pattern Signature pane.

Configuration File Pane

Description	Enables you to edit a specific configuration file that is part of a package. For example, you can edit the **portNumberToPortName.xml** file so that specific port numbers, names, or types are discovered. **To access:** Click a specific configuration file in the Discovery Resources pane.
Important Information	The following files are for internal use only and should only be changed by users with an advanced knowledge of pattern-writing: ➤ discoveryPolicy.xml ➤ jythonGlobalLibs.xml For details, see "Resource Files" on page 235 and "Internal Configuration Files" on page 238.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
🔍	Find specific text in the configuration file. For details, see "Find Text Dialog Box" on page 251.
☞	Click to go to a specific line in the configuration file. In the Go To Line dialog box, enter the line number.
📝	Click to open the file in an external editor.

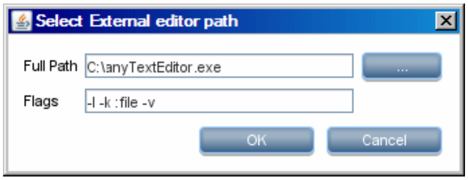
GUI Element (A–Z)	Description
	Click to edit the external editor preferences. You can run the editor by adding flags to the path. In the following example:  Full Path C:\anyTextEditor.exe Flags -l -k :file -v **:file** sets the place of the file in relation to the flags. The user cannot set the file name.
✓ XML	For XML files, signifies that the code is valid.
✗ XML	For XML files, signifies that the code is not valid.

Discovery Pattern Source Editor Window

Description	Enables you to edit a pattern script. **To access:** Right-click a pattern in the Discovery Resources pane and select **Edit Pattern Source**.
Useful Links	"Discovery Resources Pane" on page 246

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
🔍	Find specific text in the pattern script. For details, see "Find Text Dialog Box" on page 251.
	Click to go to a specific line in the pattern script. In the **Go To Line** dialog box, enter the line number.

GUI Element (A–Z)	Description
✎	Click to open the pattern script in an external text editor.
📋	Click to edit the external editor preferences. You can run the editor by adding flags to the path. In the following example:  Select External editor path Full Path: C:\anyTextEditor.exe Flags: -l -k :file -v **:file** sets the place of the file in relation to the flags. The user cannot set the file name.
✓ XML	Signifies that the code is valid.
✗ XML	Signifies that the code is invalid.

Discovery Resources Pane

Description	Enables you to locate a specific package, pattern, script, configuration file, or external resource. You can also create a pattern, Jython script, configuration file, or Discovery wizard, and you can import an external resource. **To access: Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources**
Important Information	Depending which level you select in the Discovery Resources pane, different information is displayed in the View pane. If you select: ➤ One of the following folders: Discovery Packages root, a specific package, a pattern, script, configuration file, or external resource: a list of the resources in that folder is displayed. To access a resource directly, double-click the resource in the View pane. ➤ A specific pattern: The Pattern Signature and Pattern Management panes are displayed. For details, see “Pattern Signature Tab” on page 265 and “Pattern Management Tab” on page 258. ➤ A script or configuration file: The script editor is displayed. For details, see “Script Pane” on page 273. ➤ An external resource: Information about the file is displayed.
Useful Links	“Package Manager User Interface” in *Model Management*.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✳	Click to: ➤ **Create a pattern.** Enter the pattern name and click **OK**. The new pattern is added to the << No Package >> folder. Edit the pattern. For details, see “Pattern Signature Tab” on page 265 and “Pattern Management Tab” on page 258. For details on moving a pattern to a package, see “Create Custom Package/Edit Package Wizard” in *Model Management*. ➤ **Create a Jython script.** Enter the script name. For details, see “Script Pane” on page 273. ➤ **Create a configuration file.** Enter the configuration file name. By default, the file takes an .xml extension. To give the file another extension, for example, *.properties, name the file and include the extension. Add the appropriate XML code or other content. For XML files, you can save the file only if it is valid. For details, see “Configuration File Pane” on page 243. ➤ **Import an external resource.** In the browser that opens, locate the resource to be imported and click **Open**. ➤ **Create a Discovery Wizard.** Name the new wizard. By default, the file takes an .xml extension. A new file is added to the Discovery Wizard folder of the << No Package >> folder. The file is in template format.
✖	Click to delete the resource.
🔄	Click to refresh the list of packages.
🔍	Click to open the Find Discovery Resource dialog box. For details on filtering, see “Filtering Results” on page 37.
📦	Discovery packages root. Displays a list of all packages.
📦	Package root. Displays a list of all resources included in the package. You can view any of these resources by clicking the resource in the Discovery Resources pane.

GUI Element (A–Z)	Description
<Script files>	Right-click a file to: ➤ **Save as.** Save the file under a new name. Use this option to clone an existing file. The new file includes all attributes of the existing file. Make any necessary changes to the file and save it. ➤ **Open in Frame.** Select to open the file in a new window.
<Configuration files>	Right-click a file to: ➤ **Save as.** Save the file under a new name. Use this option to clone an existing file. The new file includes all attributes of the existing file. Make any necessary changes to the file and save it. ➤ **Open in Frame.** Select to open the file in a new window.
<External resource files>	An external resource is any file needed by DDM to perform discovery. For example, the nmap.exe file is needed for credential-less discovery. ➤ Right-click a file to: ➤ **Save as.** Save the resource under a new name. Use this option to clone an existing resource. The new resource includes all attributes of the existing resource and is saved to the same location in the file system. Make any necessary changes to the new resource and save it. ➤ Select the file to display information in the View pane. You can open an external resource or export it.

GUI Element (A–Z)	Description
<Pattern files>	Right-click a file to: ➤ **Save as.** Save the pattern under a new name. Use this option to clone an existing pattern. The new pattern includes all attributes of the existing pattern. Give a name to the new pattern, and change the necessary attributes. ➤ **Go to Discovery job.** When enabled, click to open the Run Discovery window with the job selected. This option is enabled if the pattern is included in a job. ➤ **Edit pattern source.** Opens the pattern source editor where you can make changes to the pattern. For details, see "Discovery Pattern Source Editor Window" on page 244.
<Script files>	Right-click a file to: ➤ **Save as.** Save the script under a new name. Use this option to clone an existing script. The new script includes all attributes of the existing script. Make any necessary changes to the script and save it. ➤ **Open in Frame.** Select to open the script in a new window. For details on editing the script, see "Discovery Pattern Source Editor Window" on page 244.

Find Discovery Resource Dialog Box

Description	Enables you to build a search query to find a particular resource or job. **To access:** ➤ In the Discovery Modules pane, select a job and click the **Search for Discovery Job** button. ➤ In the Discovery Resources pane, select a resource and click the **Find resource** filter button.
Useful Links	"Discovery Resources Pane" on page 246

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
...	Click to select a CIT from the dialog box that opens. Click **OK** to return to the Find Discovery Resource dialog box. **Note:** This button is not accessible when **Name** is selected.
Direction	Searches forwards or backwards through the packages.
Find All	All resources meeting the search criteria are highlighted in the Discovery Resources pane.
Find Discovery resource by	Choose between: ➤ **Name.** Enter the name, or part of it, of the resources. ➤ **Pattern input type.** CIs that trigger the job. Click the button to open the Choose Configuration Item Type dialog box. Locate the CI type that you are searching for. ➤ **Pattern output type.** CIs that are discovered as a result of the activated job.
Find Next	The next resource meeting the search criteria is highlighted in the Discovery Resources pane.

Find Text Dialog Box

Description	Enables you to find text in a script or configuration file. **To access:** Select a script or configuration file and click the **Find text** button in the file pane.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Direction	Search forwards or backwards through the script or configuration file.
Find what	Type the text to be found or click the down arrow to choose from previous searches. Click the adjacent arrow to display a list of symbols you can use in wildcard or regular expression searches.
Options	Select an option to narrow your search.
Target	➤ **Global**. Searches throughout the file. ➤ **Selected Text**. Searches through the selected text.

Input TQL Editor Window

Description	Enables you to define which CIs can be Trigger CIs for jobs that run a specific pattern. **To access: Manage Discovery Resources** > select a pattern > **Pattern Signature** tab > click the **Edit** button next to the Input TQL box.
Useful Links	➤ "Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs" on page 58 ➤ "Trigger TQL Editor Window" on page 179

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<Panes>	➤ CI Type Selector Pane ➤ Editing Pane ➤ Information Pane
TQL Name	The name of the trigger TQL query that activates the job.

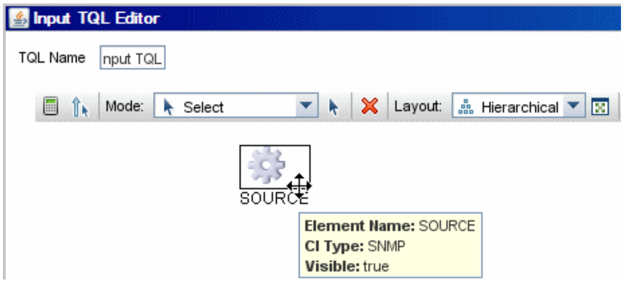
CI Type Selector Pane

Description	Displays a hierarchical tree structure of the CI Types found in the CMDB. For more details, see "CI Type Manager User Interface" in *Model Management*. **Note**: The number of instances of each CIT in the CMDB is displayed to the right of each CIT. To create or modify a TQL query, click and drag nodes to the Editing pane and define the relationship between them. Your changes are saved to the CMDB. For details, see "Add Nodes and Relationships to a TQL Query" in *Model Management*.
Included in Tasks	➤ "Define a TQL Query" on page 333 in *Model Management* ➤ "Create a Pattern View" in *Model Management*

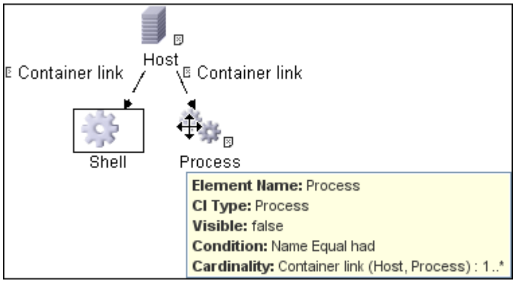
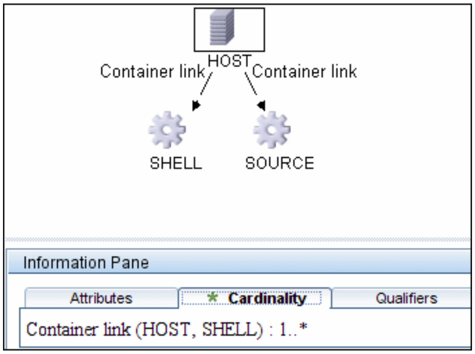
Editing Pane

Description	Enables you to edit the node.

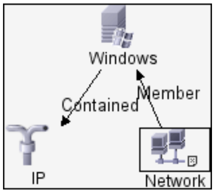
The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
<node>	Hold the cursor over a node to view information about the node:  Input TQL Editor TQL Name nput TQL Mode: Select Layout: Hierarchical SOURCE **Element Name:** SOURCE **CI Type:** SNMP **Visible:** true
<right-click menu>	For details, see “Context Menu Options” in *Model Management*
<Toolbar>	For details, see “Toolbar Options” in *Model Management*

Information Pane

Description	Displays the properties, conditions, and cardinality for the selected node and relationship.
Important Information	Hold the pointer over a node to view information:  Host Container link Container link Shell Process **Element Name:** Process **CI Type:** Process **Visible:** false **Condition:** Name Equal had **Cardinality:** Container link (Host, Process) : 1..* A small green indicator is displayed next to the tabs that include information:  HOST Container link Container link SHELL SOURCE Information Pane Attributes Cardinality Qualifiers Container link (HOST, SHELL) : 1..*

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A-Z)	Description
✎	Click **Edit** to open the relevant dialog box for the selected tab.
Attributes	Displays the attribute conditions defined for the node or the relationship. For details, see “Attribute Tab” in *Model Management*.
Cardinality	Cardinality defines how many nodes you expect to have at the other end of a relationship. For example, in a relationship between host and IP, if the cardinality is 1:3, the TQL retrieves only those hosts that are connected to between one and three IPs. For details, see “Cardinality Tab” in *Model Management*.
Details	➤ To open the Node or Relationship Properties dialog box, select a node or relationship in the Editing pane and click the **Edit** button. For details, see “Node/Relationship Properties Dialog Box” in *Model Management*. ➤ **CI Type.** The CIT of the selected node/relationship. ➤ **Visible.** A tick signifies that the selected node/relationship is visible in the topology map. When the node/relationship is not visible, a box ⊠ is displayed to the right of the selected node/relationship in the Editing pane:  ➤ **Include subtypes.** Display both the selected CI and its descendants in the topology map.
Qualifiers	Displays the qualifier conditions defined for the node or the relationship. For details, see “Qualifier Tab” in *Model Management*.

GUI Element (A-Z)	Description
Selected Identities	Displays the element instances that are used to define what should be included in the TQL results. For details, see “Identity Tab”in *Model Management*.

Manage Discovery Resources Window

Description	Enables you to view or edit default parameter values used for the DDM process. **To access: Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources** or right-click a job in the Run Discovery window.
Important Information	**Note:** An asterisk (*) next to a resource (pattern, script, or configuration file) signifies that the resource has changed since the package (in which it is included) was deployed. If the original package is redeployed, the changes are deleted from the resource. To save the changes, move the resource to a new package and deploy the package (the asterisk disappears). **Caution:** Only administrators with an expert knowledge of the DDM process should delete packages.
Useful Links	➤ “Pattern Signature Tab” on page 265 ➤ “Global Configuration Files Pane” on page 267 ➤ “Pattern Management Tab” on page 258 ➤ “Script Pane” on page 273 ➤ “Discovery Resources Pane” on page 246 ➤ “Configuration File Pane” on page 243 ➤ *Discovery and Dependency Mapping Content Guide*

Parse Rule Editor Dialog Box

Description	Enables you to create a rule that matches an attribute to process-related information (IP, port, command line, and owner). **To access:** Click **Set Attributes** in the Software Identification Rule Editor dialog box.
Important Information	Only users with a knowledge of regular expressions should make changes to a rule.
Included in Tasks	"Discover Software Elements – Scenario" on page 232
Useful Links	➤ "Attribute Editor Dialog Box" on page 240 ➤ "Software Identification Rule Editor Dialog Box" on page 276

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Process Attribute	Choose between the **Port**, **IP**, **Command Line**, or **Owner** process-related information. The rule is invoked on the attribute you choose here.
Regular Expression	Enables you to create a dynamic expression that finds at least one process that defines this software element. The regular expression is invoked on the value in the Process Attribute field. For example, a command line process includes the following regular expression: **.+\s+(\w+)$** This expression searches for any character, followed by a space or spaces, followed by a word or words (a-z or A-Z or 0-9) that appear at the end of the line. The following command line matches this regular expression: **c:\ora10\bin\oracle.exe UCMDB**

GUI Element (A–Z)	Description
Rule ID	Enter a unique name for the rule. The Rule ID is needed to identify the rule in the Software Element Additional Attributes value. For details, see “Software Element Additional Attributes” on page 275.

Pattern Management Tab

Description	Enables you to define how DDM filters the results. **To access:** Select a specific pattern in the Discovery Resources pane.
Important Information	Click the **Save** button to save any changes you make.
Useful Links	“The DiscoveryProbe.properties File” on page 39

Automatic Deletion Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description

Enable automatic deletion of removed CIs	Select this check box to automatically delete CIs that should be removed from the database if the DDM Probe does not find them during its next invocation.
	To add CITs to the list of CIs, click the **Add** button. In the Choose Configuration Item Type dialog box, choose the CITs that should be automatically deleted.
	The changes you make here are added to the pattern file, for example:
	`<resultMechanism isEnabled="true">`
	`    <autoDeleteCITs isEnabled="true">`
	`        <CIT>networkshare</CIT>`
	`    </autoDeleteCITs>`
	`</resultMechanism>`
	For details on how the DDM Probe handles CI deletion, see “Automatically Deleted System Components” on page 226.
	Note: This check box is enabled only when the **Filter unchanged results** check box in the Results Management pane is selected.

Execution Options Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Create communication logs	Choose to create a log file that logs the connection between the Probe and a remote machine. For details on how the communication logs work, see “Record DDM Code” on page 342. ➤ **Always/On Failure.** When selected, an ad-hoc request is set up to retrieve the communication log for the selected Trigger CI. **Always.** DDM always creates communication logs. **On Failure.** DDM automatically creates a log of the session in case discovery should fail (that is, DDM throws an exception or reports an error; report of a warning does not create a communication log). This is useful when you need to analyze which queries or operations take most of the time, send data for analysis from different locations, and so on. If the job completes successfully, no log is created. ➤ **Never.** No communication logs are saved to the file system or to memory. The communication log can reside in two places: in the Probe’s memory or on the file system (as a record file). When requested, DDM displays the log retrieved from the Probe’s memory. If no log exists in the memory, DDM displays the log that resides on the file system. If no log exists on the file system, a message is displayed. The communication log files are created on the Probe Manager under the **C:\hp\DDM\DiscoveryProbe \root\lib\collectors\probeManager\record** folder.

GUI Element (A–Z)	Description
Enable Automatic Deletion	Choose between: ➤ **Always**. Automatic Deletion is always enabled, regardless of whether discovery succeeds or fails. ➤ **On Success or Warnings**. Automatic Deletion is enabled only when discovery finishes with a success or warning status. In the case of a discovery error, nothing is removed. ➤ **Only on Success**. Automatic Deletion is enabled only when discovery finishes with a success status. In the case of a discovery error or warning, nothing is removed (this is the default). This element functions together with the **Enable automatic deletion of removed CIs** check box in the Automatic Deletion pane. For details on discovery status, see "Discovery Status Pane" on page 135.
Include Results in Communication Log	Select to enable capturing the discovered results with the created communication log; these discovered results may help in investigating various discovery problems.
Max. Execution Time	The maximum time allowed for a pattern to run on one Trigger CI.
Max. 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 this box empty, the 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.

General Options Pane

The following elements are included:

GUI Element (A–Z)	Description
Enable collecting 'Discovered by' data	➤ **Selected.** DDM collects data on the results of running the pattern. This data is then used to enable rediscovery of CIs. The data is necessary for the Discovery tab in IT Universe to function correctly. It is also used for the View Based Discovery Status functionality which leverages the data to aggregate the complete discovery status for certain views. ➤ **Cleared.** DDM does not collect this data. The check box needs to be cleared for patterns where rediscovery is not helpful. For example, the Range IPs by ICMP job has this check box cleared by default because its Trigger CI is the Probe Gateway and so all CIs discovered by this job have the same Trigger CI. If the check box was not cleared, a rediscovery attempt on any view containing any single IP would result in a ping sweep throughout the entire customer network, certainly not desirable behavior. The job results of this pattern are displayed in the Discovery for View dialog box only if this check box is selected. For details, see "Check Status of Application Discovery (Rediscover a View)" and "Discovery and Changes Summary Dialog Box" in *Model Management*.

Probe Selection Pane

Description	Enables you to specify which Probe to use with a pattern. **To access:** Select a specific pattern in the Discovery Resources pane.
Important Information	By default, DDM automatically chooses the Probe for the Trigger CI according to the CI's related host. After obtaining the CI's related host, DDM chooses one of the host's IPs and chooses the Probe according to the Probe's network scope definitions. This may fail in the following situations: ➤ A Trigger CI does not have a related host (such as the **network** CIT). ➤ A triggered CI's host has multiple IPs, each belonging to a different Probe. To resolve these issues, you can specify which Probe to use with the pattern by: ➤ In the Probe Selection section, selecting **Override default probe selection**. ➤ In the Probe box, typing the Probe to use for the task.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Override default probe selection	You can use calculated values such as: **${Network.network_domain}** This value uses a syntax similar to that used by Triggered CI data in the Pattern Signature tab. For details, see "Triggered CI Data Pane" on page 269.

Result Grouping Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Grouping Interval (Seconds)	To group results in the Probe before they are sent to the server, type the value that indicates how long results are stored in the Probe before being transferred to the server. **Note**: If you enter a value in both boxes, DDM applies the value of whichever occurs first.
Max. CIs in group	Specify the number of CIs that should accumulate in the Probe before being transferred to the server.

Results Management Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Enable aging	Select this check box to run the aging mechanism that specifies how long a period must pass in which CIs are discovered, before DDM treats these CIs as no longer relevant and removes them. For details on aging, see "The Aging Mechanism – Overview" in *Model Management*.
Filter unchanged results	Select this check box for the Probe to send to the CMDB only those CIs that are unchanged since the last time results were sent to the server and that answer to the filter criteria. For details on filtering, see "Filtering Results" on page 37.

Pattern Signature Tab

Description	Enables you to define a pattern by specifying: ➤ which CITs the pattern should discover ➤ which protocols are needed to perform discovery **To access:** Select a specific pattern in the Discovery Resources pane.
Included in Tasks	"Implement a Pattern" on page 317

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✎	Click the button to open the Input TQL editor. For details, see "Input TQL Editor Window" on page 251.
✖	Click to remove the Input TQL from the pattern.
Description	The description of the pattern.
Input TQL	Defines which CIs can be Trigger CIs for jobs that run this pattern. **Note:** Since this field is optional, not all patterns include an input TQL. **NA** signifies **not applicable**, that is, currently this pattern does not have an input TQL definition. Click the button to open the Input TQL Editor window. For details, see "Input TQL Editor Window" on page 251. For an explanation, see "Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs" on page 58. For an example, see "Example of Input TQL Definition" on page 319.
Trigger CIT ...	The CIT used as the trigger that activates the selected pattern. The trigger CIT is used as the pattern input. For details, see "Define Pattern Input (Trigger CIT and Input TQL)" on page 318. Click the button to choose a CIT to use as the trigger.

Discovered CITs Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to open the **Choose Discovered Class** dialog box, to select a CIT that is to be discovered by the pattern. For details, see "Choose Discovered Class Dialog Box" on page 241.
❌	Click to remove the CIT from the list of CITs that the pattern discovers.
🗺	You can choose to view a map of the CITs and links that are discovered by the pattern, instead of a list. Click the button to open the Discovered CITs Map window. The CIs and relationship links discovered by the pattern are shown.
CITs	List of CITs that the pattern discovers.

Discovery Pattern Parameters Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to open the Parameter Editor. Enter details on the parameter. The value you enter here is assigned to the attribute.
✏	Select a parameter and click the button to open the Parameter Editor and make changes.
❌	Click to remove a parameter.
Name	Each row represents the definitions for one parameter.
Value	Separate values with commas.

Global Configuration Files Pane

Description	Enables you to add default configuration files to the pattern, as well as the specific configuration files that the pattern needs. **To access:** ➤ In Manage Resources, select a pattern and the Pattern Signature tab. ➤ In Run Discovery, select a job and the Properties tab.
Important Information	The configuration file applicationsSignature.xml opens the Software Library dialog box. For details, see “Software Library Dialog Box” on page 278. The applicationsSignature.xml file contains a list of all applications that DDM attempts to find in the environment.
Included in Tasks	“Discover Software Elements – Scenario” on page 232

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to open the **Global Configuration Files** dialog box, to select configuration files that are needed by the pattern.
❌	Click to delete a selected configuration file.
✏	Select a configuration file and click to open the appropriate editor. For example, the file msServerTypes.xml opens the Script Editor.

Required Permissions Pane

Description	Enables you to view the permissions that you have configured for a pattern. **To access: Manage Discovery Resources** > select a pattern > **Pattern Signature** tab > **Required Permissions** pane.
Important Information	➤ Workflow: ➤ Configure the permissions in the Permission Editor dialog box. ➤ View the permissions in this pane. ➤ When working with jobs in the Run Discovery window, view these permissions for a specific job. ➤ For details on the fields in this pane, see “Permission Editor Dialog Box” on page 271.
Useful Links	➤ “Permission Editor Dialog Box” on page 271 ➤ “Discovery Permissions Window” on page 148 ➤ “Viewing Permissions While Running Jobs” on page 95

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
➕	Click to add a permission object. The Permission Editor dialog box opens. For details, see “Permission Editor Dialog Box” on page 271.
✖	Select a permission object and click to delete it.
✎	Select a permission object and click the button to edit. For details, see “Permission Editor Dialog Box” on page 271.
⇧ ⇩	Change the order of the permissions by selecting the permission object and clicking the up or down button. The order given here is the order in which the credentials are verified.
📄 ▾	Export a permission object in Excel, PDF, RFT, CSV, or XML format. For details, see “Browse Mode” in *Model Management*.

Required Discovery Protocols Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	Opens the **Add Required Protocol** dialog box.
✖	Click to remove an existing protocol.
Protocols	List of protocols required by the pattern for the task. For example, the NTCmd protocol, together with its user name, password, and other parameters, is needed for DDM to access a Windows system.

Triggered CI Data Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	Add Trigger CI data to the pattern.
✖	Remove Trigger CI data from the pattern.
✎	Edit the Trigger CI data in the Parameter Editor dialog box.

GUI Element (A–Z)	Description
Name	The information that is needed to perform a task on a specific CI. This information is passed to the CI queried in the task.
Value	The attribute value. Variables are written using the following syntax: ${VARIABLE_NAME.attributeName} where **VARIABLE_NAME** can be one of three predefined variables: ➤ **SOURCE**. The CI that functions as the task's trigger. ➤ **HOST**. The host in which the triggered CI is contained. ➤ **PARAMETERS**. The parameter defined in the **Parameter** section. You can create a variable. For example, ${SOURCE.network_netaddr} indicates that the trigger CI is a network.

Used Scripts Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
⇧ ⇩	Change the order of the scripts. DDM runs the scripts in the order in which they appear here.
✚	Add a script to the pattern.
✖	Remove a script from the pattern.
✎	Edit the selected script in the Script Editor that opens.
Scripts	A list of Jython scripts used by the pattern. The Jython scripts that appear in bold are the scripts that the currently selected pattern is using.

Permission Editor Dialog Box

Description	Enables you to configure a pattern you have written, so that users can view permissions for the job. **To access: Manage Discovery Resources** > select a pattern > **Pattern Signature** tab > **Required Permissions** pane > click the **Add** button.
Important Information	The information you define here is not dynamic, that is, if a pattern is changed, the information in this dialog box is not updated.
Useful Links	➤ "Discovery Permissions Window" on page 148 ➤ "Viewing Permissions While Running Jobs" on page 95 ➤ "Required Permissions Pane" on page 268 ➤ "Discovery Job Details Pane" on page 134

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Operation	The action that is being run.
Permission	Enter a name for the permission, to appear in the Required Permissions pane.
Usage Description	Free text that you enter to describe the permission object and its parameters. This text is usually a general comment on the type of permission object, whereas the description is a more specific comment. For example, you could enter Permissions for host machines here, and Permissions for host machines running on Windows for a particular row.

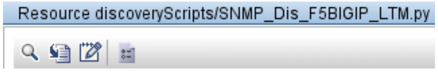
Permission Objects and Parameters Dialog Box

GUI Element (A–Z)	Description
✚	Click to open the **Permission Object and Parameter** dialog box. You can enter more than one object or parameter for each permission. The information you enter in this dialog box appears in the Required Permissions pane, in the Objects and Parameters column.
✖	Click to delete a permission object.
✎	Click to edit an existing permission object.
Context	Specific information about the permission object's environment, for example, Windows or UNIX.
Parameter	The parameters that are needed during the job run. For example, the UNIX permission object cat needs the /etc/passwd parameter.
Permission Object	The name of the command, table, or other content of the Jython script.

Script Editor Window

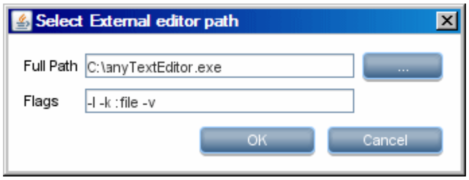
Description	Enables you to edit a specific script that is part of a package. **To access:** ➤ Right-click a script in the Discovery Resources pane and choose **Open in Frame**. ➤ Select a configuration file in the Global Configuration Files pane and click the **Edit** button. For details, see "Script Pane" on page 273.

Script Pane

Description	Enables you to edit a specific script that is part of a package. **To access:** Click a specific script in the Discovery Resources pane.
Important Information	The script pane title bar includes the actual physical location of the script. For example, the following script is located in **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryScripts** (or **probeGateway \discoveryScripts**):  Resource discoveryScripts/SNMP_Dis_F5BIGIP_LTM.py
Useful Links	Chapter 11, "Content Development and Writing"

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
🔍	Find specific text in the script. For details, see "Find Text Dialog Box" on page 251.
↪	Click to go to a specific line in a script. In the Go To Line dialog box, enter the line number.
📝	Click to open the script in an external text editor.

GUI Element (A–Z)	Description
📝	Click to edit the external editor preferences. You can run the editor by adding flags to the path. In the following example:  Select External editor path Full Path: C:\anyTextEditor.exe Flags: -l -k :file -v **:file** sets the place of the file in relation to the flags. The user cannot set the file name.
✎	Switch to simple code editor. Click to open a text editor, to use in cases where the advanced editor is causing problems.
✓ XML	For Jython files, signifies that the code is valid.
✗ XML	For Jython files, signifies that the code is not valid.
Fix validation errors	See Validation Information below. **Note**: This button is displayed when a script contains Framework API errors.
<script>	The Jython script used by the package. For details on working with Jython, see "Step 3: Create Jython Code" on page 328.
Validation Information	If a script is not valid, Validation Information displays the errors in the script, for example: Script has failed validation. At line 48: Factory.getProtocolProperty(found. This is a problem - Usage of Factory is deprecated. Use Framework.getProtocolProperty instead. Click **Fix validation errors** then **OK** to update the script. The error may occur due to changes in the Framework object's API. For details, see "Discovery and Dependency Mapping API Changes" in *Reference Information*.

Software Element Attribute Assignment Rules Dialog Box

Description	Enables you to define a regular expression that discovers a specific software element according to a CIT's attribute value. **To access:** Click **Set Attributes** in the Software Identification Rule Editor dialog box.
Included in Tasks	"Discover Software Elements – Scenario" on page 232
Useful Links	➤ "Parse Rule Editor Dialog Box" on page 257 ➤ "Attribute Editor Dialog Box" on page 240 ➤ "Software Identification Rule Editor Dialog Box" on page 276 ➤ "The Software Element CIT" on page 63

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
✚	Click to add a regular expression that determines the attribute of the CI to be discovered, or to add an attribute.
✖	Click to delete the regular expression or the attribute.
✎	Click to edit an existing regular expression or attribute.
Parse Rules	For details, see "Parse Rule Editor Dialog Box" on page 257.
Software Element Additional Attributes	For details, see "Attribute Editor Dialog Box" on page 240.

Software Identification Rule Editor Dialog Box

Description	Enables you to define a new software element. **To access:** In the Software Library dialog box, click the **Add** button or select an existing element and click the **Edit** button.
Important Information	Each parse rule must be matched by at least one process.
Included in Tasks	"Discover Software Elements – Scenario" on page 232
Useful Links	➤ "Global Configuration Files Pane" on page 267 ➤ "The Software Element CIT" on page 63

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Set Attributes	Click to add attributes to the component. For details, see "Software Element Attribute Assignment Rules Dialog Box" on page 275.
✚	Click to add a process or a software configuration file.
✖	Select a process or a software configuration file and click to delete.
✎	Select a process or a software configuration file and click to edit.
Category	You can: ➤ Choose the category under which the new software element should appear. ➤ Change the category for an existing element. ➤ Add a new category by typing its name in this field. The changes you make here are immediately displayed in the Software Library dialog box.
CI Type	Select the CIT that is to be discovered.

GUI Element (A–Z)	Description
Identifying Configuration Files	To add a configuration file, click the **Add** button to open the Configuration File Name dialog box. Enter the full path to the software element's configuration file and the file name.
Identifying Processes	To add a process that can identify a specific software element, click the **Add** button. The **Process Data** dialog box opens: **Key**. Select this check box if, during discovery, DDM must distinguish between applications that run similar processes (IP, port, command line, or owner). For an explanation of this box, see "Identifying Software Element Processes" on page 229. **Name**. Enter the exact name of the process, for example, **java.exe**. **Port**. Add a port number or name, either by typing a number or by clicking the **Add** button then selecting the ports in the Global Ports List. ➤ If the process has to listen at a specific port, the port should be listed. You can enter more than one port, separated by commas, for example, **8888,8081,8080, 81,8000,82,80**. ➤ If the process does not have to listen at a specific port (that is, the software element can use any port), leave this field empty. **Port match is optional.** ➤ Select this check box to enable discovery of processes that are not listening on any of the ports entered in the Port field (that is, identification is by process name only). If you select this check box, the word **optional** is added to the Port field. ➤ Clear this check box to enable discovery of processes based on process name and the port number entered in the Port field. **Command Line**. The software element can also be mapped using the process name. In this case, you must add a process command line (or part of it) with which the process name uniquely identifies the software, for example, **c:\ora10\bin\oracle.exe UCMDB**.

GUI Element (A–Z)	Description
Produced Software Name	The name of the software element to be created by this signature.
Software Signature Name	The name of the definition. **Note**: This is not the software element's name but a name you give to differentiate this discovery from similar discoveries.

Software Library Dialog Box

Description	Enables you to view the logical groups of software elements. **To access:** ➤ **Run Discovery window** > **Host Resources and Application Dependency module** > **Software Element CF by Shell job**. Locate the **Global Configuration Files** pane in the Properties tab. Select **applicationsSignature.xml** and click the **Edit** button. ➤ **Manage Discovery Resources window** > **Host_Resources_Basic package** > **Dis_AppComponents_CF pattern**. Locate the **Global Configuration File**s pane in the Pattern Signature tab. Select **applicationsSignature.xml** and click the **Edit** button. ➤ In the Infrastructure Wizard Preferences page, open the **Choose Software Element to be discovered** box.
Important Information	The software elements are organized in logical categories. You can change the names of these elements, you can move an element to another category, and you can define new elements and categories. For details, see the **Category** entry in "Software Identification Rule Editor Dialog Box" on page 276. The code you define in this dialog box and the Software Element Editor dialog box overwrites the code in applicationsSignature.xml.

Included in Tasks	"Discover Software Elements – Scenario" on page 232
Useful Links	➤ "Global Configuration Files Pane" on page 267 ➤ "The Software Element CIT" on page 63

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
⊞ ☑ 📁 ⊞ ☑ 📁	Select a check box to include a category or software element in the discovery. Clear a check box to remove the category or element from the discovery.
✚	Click to define a new software element. For details, see "Software Identification Rule Editor Dialog Box" on page 276.
✖	Select a software element and click to delete the element.
✎	Select a software element and click to make changes to the element. For details, see "Software Identification Rule Editor Dialog Box" on page 276.
<List of software elements>	List of objects that are software elements.

9

Show Status Snapshot

This chapter provides information on viewing the current status of the discovered CIs in the DDM Probes.

This chapter includes:

Concepts

- Show Status Snapshot – Overview on page 281

Tasks

- View Current Status of Discovered CIs on page 282

Reference

- Show Status Snapshot User Interface on page 282

Show Status Snapshot – Overview

You use Show Status Snapshot to view the current status of the discovered CIs in the Probes. Show Status Snapshot retrieves the status from the Probes and displays the results in a view.

The view is not automatically updated; to refresh the status data, click the **Get snapshot** button.

View Current Status of Discovered CIs

This task describes how to view the current status of discovered CIs.

This task includes the following steps:

- “Prerequisites” on page 282
- “Access Show Status Snapshot” on page 282

1 Prerequisites

Verify that the Probe is enabled and is connected to the HP Business Availability Center Gateway server. For details, see “Get Started With the DDM Probe” on page 38.

2 Access Show Status Snapshot

a Go to **Discovery** > **Show Status Snapshot**.

b Select a connected probe.

c Click the **Get Snapshot** button.

d Select jobs from the Progress list and click the **View Job progress** button. The Job Progress window opens.

Show Status Snapshot User Interface

This section describes:

- [Job Name] Dialog Box on page 283
- Show Status Snapshot Window on page 284

[Job Name] Dialog Box

Description	Enables you to view details about a job, including its scheduling, as well as job statistics. **To access:** Select a job in the Progress pane of the Show Status Snapshot window and click the **View job progress** button.
Important Information	Double-click a job to open a further dialog box with details of the job.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Job Details	**Status.** Can be **Scheduled** (the job runs according to a defined schedule) or **Running** (the job is running now). **Last updated.** The last time that the job was updated. **Threads.** The number of threads currently allocated to this job. **Progress.** The number of Trigger CIs in the job and the number of Trigger CIs that the Probe has finished working on.
Schedule	**Previous invocation.** The last time that DDM ran the job. **Next invocation.** The next time that DDM is scheduled to run the job. **Last duration.** The length of time, in seconds, taken to run the job in the previous invocation. **Average duration.** The average duration, in seconds, of the time it took the Probe to run this job. **Recurrence.** The number of times a job is run in a week. For example, if a job is scheduled to run daily, it runs 7 times in a week. If a job is scheduled to run weekly, Recurrence = 1.
Statistics Results	For details, see “Statistics Results Pane” on page 286.

Show Status Snapshot Window

Description	Enables you to view the current status of discovered CIs and all active jobs running on the Probes. **To access: Admin** > **Universal CMDB** > **Discovery** > **Show Status Snapshot**
Important Information	Depending on what you select in the Domains and Probes pane, different information is displayed in the View pane. If you select: ➤ a domain, you can view details and CIT statistics for the domain. For details, see "Details Pane" on page 195 and "Statistics Results Pane" on page 286. ➤ a Probe, you can view details on the Probe (such as the Probe IP), the progress of a job and you can view CIT statistics. For details, see "Details Pane" on page 284, "Progress Pane" on page 285, "Statistics Results Pane" on page 286, and "View Pane" on page 287.
Included in Tasks	"View Current Status of Discovered CIs" on page 282
Useful Links	"Show Status Snapshot – Overview" on page 281

Details Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
Domain Type	**Customer.** A private domain used for your site. You can define several domains and each domain can include multiple Probes. Each Probe can include IP ranges but the customer domain itself has no range definition. **External.** Internet/public domain. A domain which is defined with a range. The external domain may contain only one Probe whose name equals the domain name. However, you can define several external domains in your system. For details on defining domains, see "Add New Domain Dialog Box" on page 193.

Progress Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
ⓘ	Select a CI and click this icon to view details of a job. For details, see "[Job Name] Dialog Box" on page 283.
Job	The name of the job. Double click a job to open a dialog box displaying job details. For details, see "[Job Name] Dialog Box" on page 283.
Next invocation	The next time that the Probe is scheduled to run.
Previous invocation	The last time that the Probe ran.
Progress	Can be either Scheduled or Running. If a job is running, a progress bar displays the percentage finished.
Thread count	The number of threads currently allocated to this job.
Triggered CIs	The number of CIs triggered in the job.

Statistics Results Pane

Description	Enables you to view details and CIT statistics. **To access:** Click the Default Domain or Probe name in the Domains Browser pane.

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
🔄	Click to retrieve the latest data from the Probe (data is not automatically updated).
▽	Set the time range for which to display statistics about the CITs. ➤ **All.** Displays statistics for all job runs. ➤ **Last Hour/Day/Week/Month.** Choose a period of time for which to display statistics about the CITs. ➤ **Custom Range.** Click to open the Customize Statistics Time Range dialog box. Enter the date or click the arrow to choose a date and time from the calendar, for the **To** and **From** dates. To delete a date, click **Reset**.
<Column title>	Click a column title to change the order of the CITs from ascending to descending order, or vice versa.
<right-click a title>	Choose from the following options: ➤ **Hide Column.** Select to hide a specific column. ➤ **Show All Columns.** Displayed when a column is hidden. ➤ **Customize.** Select to display or hide columns and to change the order of the columns in the table. Opens the Columns dialog box. ➤ **Auto-resize Column.** Select to change a column width to fit the contents. For details, see “Select Columns Dialog Box” in *Reference Information*.

GUI Element (A–Z)	Description
CIT	The name of the discovered CIT.
Created	The number of CIT instances created by the Probe.
Deleted	The number of CIT instances deleted by the Probe.
Discovered CIs	The sum of all the CIs for all the invocations.
Filter	The time range set with the Set Filter button.
Last updated	The date and time that the statistics table has been updated for a particular Probe.
Total	The total number of CIs in each column.
Updated	The number of CIT instances that have been updated.

View Pane

The following elements are included (unlabeled GUI elements are shown in angle brackets):

GUI Element (A–Z)	Description
📷	Click to view the current status of the discovered CIs and jobs on the selected Probe.
Last updated	The date and time at which the Get snapshot button was last pressed (that is, the date and time of the data displayed in Status Snapshot).
Probe IPs	The IP addresses defined for the Probe.
Running jobs	The number of jobs running on the Probe.
Scheduled jobs	The number of jobs that are scheduled to run according to the settings in the Discovery Scheduler. For details, see “Discovery Scheduler Dialog Box” on page 149.
Status	The status of the Probe (either disconnected or connected).
Threads	The sum of all threads currently allocated to the running jobs.

10

The DDM Web Service API

This chapter includes:

Concepts

Tasks

Reference

DDM Web Service API Overview

This chapter explains how third-party or custom tools can use the HP Discovery and Dependency Mapping Web Service to manage Discovery and Dependency Mapping (DDM).

For full documentation on the available operations, see *HP Discovery and Dependency Mapping Schema Reference*. These files are located in the following folder:

\\<HP Business Availability Center Gateway Server root directory>\AppServer\webapps\site.war\amdocs\eng\doc_lib\Discovery_and_Dependency_Mapping\DDM_Schema\webframe.html

Conventions

This chapter uses the following conventions:

- This style, Element, indicates that an item is an entity in the database or an element defined in the schema, including structures passed to or returned by methods. Plain text indicates that the item is being discussed in a general context.
- DDM elements and method arguments are spelled in the case in which they are specified in the schema. This usually means that a class name or generic reference to an instance of the class is capitalized. An element or argument to a method is not capitalized. For example, a credential is an element of type Credential passed to a method.

The HP Discovery and Dependency Mapping Web Service

The HP Discovery and Dependency Mapping Web Service is an API used to integrate applications with HP Universal CMDB (UCMDB). The API provides methods to:

- Manage credentials: view, add, update, and remove
- Manage jobs: view status, activate, and deactivate
- Manage probe ranges: view. add, and update
- Manage triggers: Add or remove a trigger CI, and add, remove, or disable a trigger TQL
- View general data on domains and probes

Users of the HP Discovery and Dependency Mapping Web Service should be familiar with:

- The SOAP specification
- An object-oriented programming language such as C++, C# or Java
- HP Universal CMDB
- Discovery and Dependency Mapping

Permissions

The administrator provides login credentials for connecting with the Web service. The required credentials depend on whether you are using DDM with a standalone version of HP Universal CMDB or from within HP Business Availability Center.

When permissions are assigned through DDM, the permission levels are View, Update, and Execute. When they are assigned using the HP Business Availability Center, the levels are View and Update, where Update also includes Execution. To view the permissions required for each operation, see each operation's request documentation in the *HP Discovery and Dependency Mapping Schema Reference*.

To assign permissions:

- **DDM with HP Universal CMDB standalone.** Log in using the credentials of a DDM user who has been granted permissions on the discovery resources.
- **DDM embedded in HP Business Availability Center.** Log in using the credentials of a Business Availability Center user. The user must have been granted the relevant permissions on the UCMDB Web Service resource in Business Availability Center.

Call the Web Service

The HP Discovery and Dependency Mapping Web Service enables calling server-side methods using standard SOAP programming techniques. If the statement cannot be parsed or if there is a problem invoking the method, the API methods throw a SoapFault exception. When a SoapFault exception is thrown, the service populates one or more of the error message, error code, and exception message fields. If there is no error, the results of the invocation are returned.

SOAP programmers can access the WSDL at:

http://<server>[:port]/axis2/services/DiscoveryService?wsdl

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

The URL for calling the service is:

http://<server>[:port]/axis2/services/DiscoveryService

Discovery and Dependency Mapping Methods

This section contains a list of the Web service operations and a brief summary of their use. For full documentation of the request and response for each operation, see *HP Discovery and Dependency Mapping Schema Reference*.

This section includes the following topics:

- "Managing Discovery Job Methods" on page 292
- "Managing Trigger Methods" on page 293
- "Domain and Probe Data Methods" on page 293
- "Credentials Data Methods" on page 294
- "Data Refresh Methods" on page 294

Managing Discovery Job Methods

- **activateJob**

  Activates the specified job.

- **deactivateJob**

  Deactivates the specified job.

- **dispatchAdHocJob**

  Dispatches a job on the probe ad-hoc. The job must be active and contain the specified trigger CI.

- **getDiscoveryJobsNames**

  Returns the list of job names.

- **isJobActive**

  Checks whether the job is active.

Managing Trigger Methods

- **addTriggerCI**

  Adds a new trigger CI to the specified job.

- **addTriggerTQL**

  Adds a new trigger TQL to the specified job.

- **disableTriggerTQL**

  Prevents the TQL from triggering the job, but does not permanently remove it from the list of queries that trigger the job.

- **removeTriggerCI**

  Removes the specified CI from the list of CIs that trigger the job.

- **removeTriggerTQL**

  Removes the specified TQL from the list of queries that trigger the job.

- **setTriggerTQLProbesLimit**

  Restrict the probes in which the TQL is active in the job to the specified list.

Domain and Probe Data Methods

- **getDomainType**

  Returns the domain type.

- **getDomainsNames**

  Returns the names of the current domains.

- **getProbeIPs**

  Returns the IP addresses of the specified probe.

- **getProbesNames**

  Returns the names of the probes in the specified domain.

- **getProbeScope**

  Returns the scope definition of the specified probe.

- **isProbeConnected**

  Checks whether the specified probe is connected.

- **updateProbeScope**

  Sets the scope of the specified probe, overriding the existing scope.

Credentials Data Methods

- **addCredentialsEntry**

  Adds a credentials entry to the specified protocol for the specified domain.

- **getCredentialsEntriesIDs**

  Returns the IDs of the credentials defined for the specified protocol.

- **getCredentialsEntry**

  Returns the credentials defined for the specified protocol. Encrypted attributes are returned empty.

- **removeCredentialsEntry**

  Removes the specified credentials from the protocol.

- **updateCredentialsEntry**

  Sets new values for properties of the specified credentials entry.

Data Refresh Methods

- **rediscoverCIs**

  Locates the triggers that discovered the specified CI objects and reruns those triggers.

  rediscoverCIs runs asynchronously. Call **checkDiscoveryProgress** to determine when the rediscovery is complete.

- **checkDiscoveryProgress**

  Returns the progress of the most recent **rediscoverCIs** call on the specified IDs. The response is a value from 0 to 1. When the response is 1, the **rediscoverCIs** call has completed.

➤ **rediscoverViewCIs**

Locates the triggers that created the data to populate the specified view, and reruns those triggers.

rediscoverViewCIs runs asynchronously. Call **checkViewDiscoveryProgress** to determine when the rediscovery is complete.

➤ **checkViewDiscoveryProgress**

Returns the progress of the most recent **rediscoverViewCIs** call on the specified view. The response is a value from 0-1. When the response is 1, the **rediscoverCIs** call has completed.

Part IV

Content Writing

11

Content Development and Writing

This chapter describes the approaches, methodologies, and practices of developing new Discovery and Dependency Mapping (DDM) content.

This chapter includes:

Concepts

Tasks

- Work with Discovery Analyzer on page 353

Reference

- Discovery and Dependency Mapping Code on page 359
- Jython Libraries and Utilities on page 362
- Job and Pattern XML Formats on page 366

Content Development and Writing Overview

Prior to beginning actual planning for development of new content, it is important for you to understand the processes and interactions commonly associated with this development.

The following sections can help you understand what you need to know and do, to successfully manage and execute a discovery development project.

This chapter:

- Assumes a working knowledge of HP Business Availability Center and some basic familiarity with the elements of the DDM system. It is meant to assist you in the learning process and does not provide a complete guide.
- Covers the stages of planning, research, and implementation of new discovery content for HP Business Availability Center using DDM, along with guidelines and considerations that need to be taken into account.
- Provides information on the key APIs of the Discovery and Dependency Mapping Framework. For full documentation on the available APIs, see the *HP Discovery and Dependency Mapping API Reference*. (Other non-formal APIs exist but even though used on out of the box patterns, they may be subject to change.)

Associating Business Value with Discovery Development

The use case for developing new discovery content should be driven by a business case and plan to produce business value. That is, the goal of mapping system components to CIs and adding them to the CMDB is to provide business value.

The content may not always be used for application mapping, although this is a common intermediate step for many use cases. Regardless of the end usage of the content, your plan should answer these questions of this approach:

- Who is the consumer? How should the consumer act on the information provided by the CIs (and the relationships between them)? What is the business context in which the CIs and relationships are to be viewed? Is the consumer of these CIs a person or a product or both?
- Once the perfect combination of CIs and relationships exists in the CMDB, how do I plan on using them to produce business value?
- What should the perfect mapping look like?
    - What term would most meaningfully describe the relationships between each CI?
    - What types of CIs would be most important to include?
    - What is the end usage and end user of the map?
- What would be the perfect report layout?

Once the business justification is established, the next step is to embody the business value in a document. This means picturing the perfect map using a drawing tool and understanding the impact and dependencies between CIs, reports, how changes are tracked, what change is important, monitoring, compliance, and additional business value as required by the use cases.

This drawing (or model) is referred as the **blueprint**.

For example, if it is critical for the application to know when a certain configuration file has changed, the file should be mapped and linked to the appropriate CI (to which it relates) in the drawn map.

Work with an SME (Subject Matter Expert) of the area, who is the end user of the developed content. This expert should point out the critical entities (CIs with attributes and relationships) that must exist in the CMDB to provide business value.

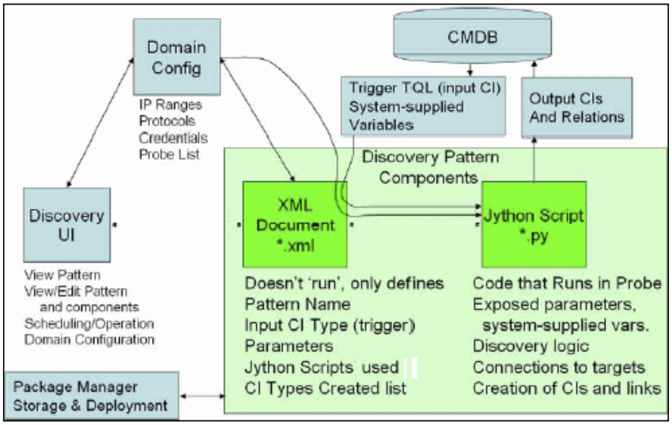
One method could be to provide a questionnaire to the application owner (also the SME in this case). The owner should be able to specify the above goals and blueprint. The owner must at least provide a current architecture of the application.

You should map critical data only and no unnecessary data: you can always enhance DDM later. The goal should be to set up a limited discovery that works and provides value. Mapping large quantities of data gives more impressive maps but can be confusing and time consuming to develop.

Once the model and business value is clear, continue to the next stage. This stage can be revisited as more concrete information is provided from the next stages.

DDM Patterns and Related Components

The following diagram shows a pattern's components and the components they interact with to execute discovery. The components in green are the actual patterns, and the components in blue are components that interact with patterns.



Note that the minimum notion of a pattern is two files: an XML document and a Jython script. The DDM Framework, including input CIs, credentials, and user-supplied libraries, is exposed to the pattern at run time. Both discovery pattern components are administered through the DDM application. They are stored operationally in the CMDB itself; although the external package remains, it is not referred to for operation. The Package Manager enables preservation of the new discovery content capability.

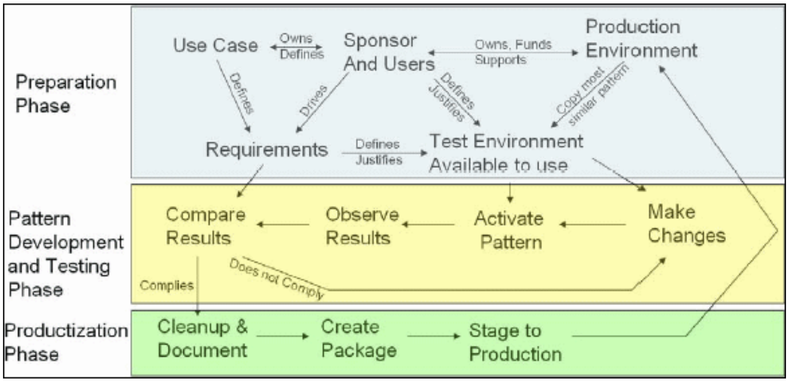
Input CIs to the pattern are provided by a TQL, and are exposed to the pattern script in system-supplied variables. Pattern parameters are also supplied as destination data, so you can configure the pattern's operation according to a pattern's specific function.

The DDM application is used to create and test new patterns. You use the job, resource, and domain configuration pages during content writing.

Patterns are stored and transported as packages. The Package Manager application and the JMX console are used to create packages from newly created patterns, and to deploy patterns on new systems.

The DDM Development Cycle

The following illustration shows a flowchart for content writing. Most of the time is spent in the middle section, which is the iterative loop of development and testing.



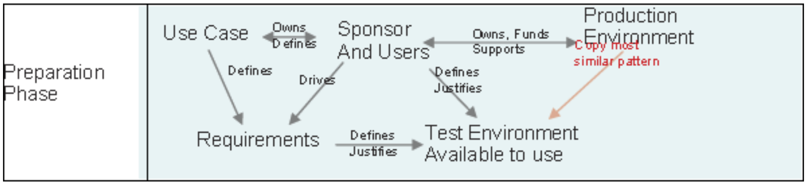
Each phase of pattern development builds on the last one.

Once you are satisfied with the way the pattern looks and works, you are ready to package it. Using either the UCMDB Package Manager or manual exporting of the components, create a DDM package *.zip file. As a best practice, you should deploy and test this package on another UCMDB system before releasing it to production, to ensure that all the components are accounted for and successfully packaged. For details on packaging, see "Create a Custom Package" in *Model Management*.

The following sections expand on each of the phases showing the most critical steps and best practices:

- Research and Preparation Phase
- Pattern Development and Testing
- Pattern Packaging and Productization

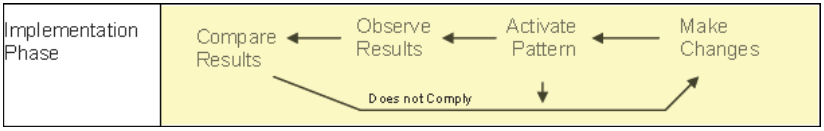
Research and Preparation Phase



The Research and Preparation phase encompasses the driving business needs and use cases, and also accounts for securing the necessary facilities to develop and test the pattern.

1. When planning to modify an existing pattern, the first technical step is to make a backup of that pattern and ensure you can return it to its pristine state. If you plan to create a new pattern, copy the most similar pattern and save it under an appropriate name. For details, see "<Pattern files>" on page 249.
2. Research how the pattern should collect data.
   - Use External tools/protocols to obtain the data
   - Develop how the pattern should create CIs based on the data
   - You now know what a similar pattern should look like

3. Determine most similar pattern based on:
   - Same CIs created
   - Same Protocols used (SNMP)
   - Same kind of targets (by OS type, versions, and so on)
4. Copy entire package.
5. Unzip into work space and rename the pattern (XML) and Jython (.py) files.



Pattern Development and Testing

The Pattern Development and Testing phase is a highly iterative process. As the pattern begins to take shape, you begin testing against the final use cases, make changes, test again, and repeat this process until the pattern complies with the requirements.

Startup and Preparation of Copy

- Modify XML parts of the pattern: Name (id) in line 1, Created CI Types, and Called Jython script name.
- Get the copy running with identical results to the original pattern.
- Comment out most of the code, especially the critical result-producing code.

Development and Testing

- Use other sample code to develop changes
- Test pattern by running it
- Use a dedicated view to validate complex results, search to validate simple results

Pattern Packaging and Productization

The Pattern Packaging and Productization phase accounts for the last phase of development. As a best practice, a final pass should be made to clean up debugging remnants, documents, and comments, to look at security considerations, and so on, before moving on to packaging. You should always have at least a readme document to explain the inner workings of the pattern. Someone (maybe even you) may need to look at this pattern in the future and will be aided greatly by even the most limited documentation.

Cleanup and Document

- Remove debugging
- Comment all functions and add some opening comments in the main section
- Create sample TQL and view for the user to test

Create Package

- Export patterns, TQL, and so on with the Package Manager. For details, see "Package Manager" in *Model Management*.
- Check any dependencies your package has on other packages, for example, if the CIs created by those packages are input CIs to your pattern.
- Use Package Manager to create a package zip. For details, see "Package Manager" in *Model Management*.
- Test deployment by removing parts of the new content and redeploying, or deploying on another test system.

DDM and Integration

DDM discovery patterns are capable of integration with other products. Consider the following definitions:

- DDM collects specific content from many targets.
- Integration collects multiple types of content from one system.

Note that these definitions do not distinguish between the methods of collection. Neither does DDM. The process of developing a new pattern is the same process for developing new integration. You do the same research, make the same choices for new vs. existing patterns, write the patterns the same way, and so on. Only a few things change:

- The final pattern's scheduling. Integration patterns may run more frequently than discovery, but it depends on the use cases.
- Input CIs:
  - Integration: non-CI trigger to run with no input: a file name or source is passed through the pattern parameter.
  - Discovery: uses regular, UCMDB CIs for input.

For integration projects, you should almost always reuse an existing pattern. The direction of the integration (from HP Business Availability Center to another product, or from another product to HP Business Availability Center) may affect your approach to development. There are field packages available for you to copy for your own uses, using proven techniques.

From HP Business Availability Center to another project:

- Create a TQL that produces the CIs and relations to be exported.
- Use a generic wrapper pattern to execute the TQL and write the results to an XML file for the external product to read.

Note: For examples of field packages, contact HP Software Support.

To integrate another product to HP Business Availability Center: Depending on how the other product exposes its data, the integration pattern acts differently:

Integration Type	Reference Example to Be Reused
Access the product's database directly	HP ED
Read in a csv or xml file produced by an export	HP ServiceCenter
Access a product's API	BMC Atrium/Remedy

Research Stage

The prerequisite of this stage is a **blueprint** of the CIs and relationships needed to be discovered by DDM, which should include the attributes that are to be discovered. For details, see "Content Development and Writing Overview" on page 300.

This section includes the following topics:

- "Modifying an Existing Pattern" on page 309
- "Writing a New Pattern" on page 309
- "Model Research" on page 310
- "Technology Research" on page 310
- "Guidelines for Choosing Ways to Access Data" on page 311
- "Summary" on page 312

Modifying an Existing Pattern

You modify an existing pattern when an out-of-the-box or field DDM pattern exists, but:

- it does not discover specific attributes that are needed
- a specific type of target (OS) is not being discovered or is being incorrectly discovered
- a specific relationship is not being discovered or created

If an existing pattern does some, but not all, of the job, your first approach should be to evaluate the existing patterns and verify if one of them almost does what is needed; if it does, you can modify the existing pattern.

You should also evaluate if an existing field pattern is available. Field patterns are discovery patterns that are available but are not out-of-the-box. Contact HP Software Support to receive the current list of field patterns.

Writing a New Pattern

A new pattern needs to be developed:

- When it is faster to write a pattern than to insert the information manually into the CMDB (generally, from about 50 to 100 CIs and relationships) or it is not a one-time effort.
- When the need justifies the effort.
- If out of the box or field patterns are not available.
- If the results can be reused.
- When the target environment or its data is available (you cannot discover what you cannot see).

Model Research

- Browse the CMDB class model (CI Type Manager) and match the entities and relations from your **blueprint** to existing CITs. It is highly recommended to adhere to the current model to avoid possible complications during version upgrade. If you need to extend the model, you should create new CITs since an upgrade may overwrite out of the box CITs.
- If some entities, relations, or attributes are lacking from the current model, you should create them. It is preferable to create a package with these CITs (which will also later hold all the discovery, views, and other artifacts relating to this package) since you need to be able to deploy these CITs on each installation of HP Business Availability Center.

Technology Research

Once you have verified that the CMDB hold the relevant CIs, the next stage is to decide how to retrieve this data from the relevant systems.

Retrieving data usually involves using a protocol to access a management part of the application, actual data of the application, or configuration files or databases that are related to the application. Any data source that can provide information on a system is valuable. Technology research requires both extensive knowledge of the system in question and sometimes creativity.

For home-grown applications, it may be helpful to provide a questionnaire form to the application owner. In this form the owner should list all the areas in the application that can provide information needed for the blueprint and business values. This information should include (but does not have to be limited to) management databases, configuration files, log files, management interfaces, administration programs, Web services, messages or events sent, and so on.

For off-the-shelf products, you should focus on documentation, forums, or support of the product. Look for administration guides, plug-ins and integrations guides, management guides, and so on. If data is still missing from the management interfaces, read about the configuration files of the application, registry entries, log files, NT event logs, and any artifacts of the application that control its correct operation.

Guidelines for Choosing Ways to Access Data

Relevance: Select sources or a combination of sources that provide the most data. If a single source supplies most information whereas the rest of the information is scattered or hard to access, try to assess the value of the remaining information by comparison with the effort or risk of getting it. Sometimes you may decide to reduce the blueprint if the value or cost does not warranty the invested effort.

Reuse: If HP Business Availability Center already includes a specific connection protocol support it is a good reason to use it. It means the DDM Framework is able to supply a ready made client and configuration for the connection. Otherwise, you may need to invest in infrastructure development. You can view the currently supported HP Business Availability Center connection protocols: **Discovery** > **Setup Discovery Probe** > **Domains and Probes pane**. For details, see "Domains and Probes Pane" on page 200.

You can add new protocols by adding new CIs to the model. For details, contact HP Software Support.

Note: To access Windows Registry data, you can use either WMI or NTCmd.

Security: Access to information usually requires credentials (user name, password), which are entered in the CMDB and are kept secure throughout the product. If possible, and if adding security does not conflict with other principles you have set, choose the least sensitive credential or protocol that still answers access needs. For example, if information is available both through JMX (standard administration interface, limited) and Telnet, it is preferable to use JMX since it inherently provides limited access and (usually) no access to the underlying platform.

Comfort: Some management interfaces may include more advanced features. For example, it might be easier to issues queries (SQL, WMI) than to navigate information trees or build regular expressions for parsing.

Developer Audience: The people who will eventually develop DDM may have an inclination towards a certain technology. This can also be considered if two technologies provide almost the same information at an equal cost in other factors.

Summary

The outcome of this stage is a document describing the access methods and the relevant information that can be extracted from each method. The document should also contain a mapping from each source to each relevant blueprint data.

Each access method should be marked according to the above instructions. Finally you should now have a plan of which sources to discover and what information to extract from each source into the blueprint model (which should by now have been mapped to the corresponding UCMDB model).

Separating Patterns

Technically, an entire discovery could be defined in a single pattern. But good design demands that a complex system be separated into simpler, more manageable components.

The following are guidelines and best practices for dividing the DDM process:

- Discovery should be done in stages. Each stage should be represented by a pattern that should map an area or tier of the system. Patterns should rely on the previous stage or tier to be discovered, to continue discovery of the system. For example, Pattern A is triggered by an application server TQL result and maps the application server tier. As part of this mapping, a JDBC connection component is mapped. Pattern B registers a JDBC connection component as a trigger TQL and uses the results of pattern A to access the database tier (for example, through the JDBC URL attribute) and maps the database tier.
- **The two-phase connect paradigm**: Most systems require credentials to access their data. This means that a user/password combination needs to be tried against these systems. The DDM administrator supplies credentials information in a secure way to the system and can give several, prioritized login credentials. This is referred to as the **Protocol Dictionary**. If the system is not accessible (for whatever reason) there is no point in performing further discovery. If the connection is successful, there needs to be a way to indicate which credential set was successfully used, for future discovery access.

These two phases lead to a separation of the two patterns in the following cases:

  - **Connection Pattern**: This is a pattern that accepts an initial trigger and looks for the existence of a remote agent on that trigger. It does so by trying all entries in the Protocol Dictionary which match this agent's type. If successful, this pattern provides as its result a remote agent CI (SNMP, WMI, and so on), which also points to the correct entry in the Protocol Dictionary for future connections. This agent CI is then part of a trigger for the content pattern.
  - **Content Pattern**: This pattern's precondition is the successful connection of the previous pattern (preconditions specified by the TQLs). These types of patterns no longer need to look through all of the Protocol Dictionary since they have a way to obtain the correct credentials from the remote agent CI and use them to log in to the discovered system.

- Different scheduling considerations can also influence discovery division. For example, a system may only be queried during off hours, so even though it would make sense to join the pattern to the same pattern discovering another system, the different schedules mean that you need to create two patterns.
- Discovery of different management interfaces or technologies to discover the same system should be placed in separate patterns. This is so that you can activate the access method appropriate for each system or organization. For example, some organizations have WMI access to machines but do not have SNMP agents installed on them.

Using External Java JAR Files Within Jython

When developing new Jython scripts, external Java Libraries (JAR files) or third party executable files are sometimes needed as either Java utility archives, connection archives such as JDBC Driver JAR files, or executable files (for example, **nmap.exe** is used for credential-less discovery).

These resources should be bundled in the package under the External Resources folder. Any resource put in this folder is automatically sent to any Probe that connects to your HP Universal CMDB server.

In addition, when discovery is launched, any JAR file resource is loaded into the Jython's classpath, making all the classes within it available for import and use.

HP Discovery and Dependency Mapping API Reference

For full documentation on the available APIs, see *HP Discovery and Dependency Mapping API Reference*. These files are located in the following folder:

\\<HP Business Availability Center Gateway Server root directory>\
AppServer\webapps\site.war\amdocs\eng\doc_lib\
Discovery_and_Dependency_Mapping\DDM_JavaDoc\index.html

Using Discovery Analyzer to Debug Content

The Discovery Analyzer tool is intended for debugging purposes when developing packages, scripts, or any other content. The tool runs a job against a remote destination and returns logs containing information, warning, and error details and results of discovered CIs.

This section includes the following topics:

- "Tasks and Records" on page 316
- "Logs" on page 316

Tasks and Records

A task file contains data regarding a task to be executed. The task consists of information such as the job's name and required parameters that define the trigger CI, for example, the remote destination address.

A record file contains task information as well as the results of a specific execution. An execution is the detailed communication (including a response) between the Probe or Discovery Analyzer (whichever module executed the task) and the remote destination.

A task that is defined by a task file can be executed against a remote destination, whereas a task that is defined by a record file (that contains extra data regarding a specific execution) can be executed and can also be played back (that is, can reproduce the same execution documented in the record file).

Logs

Logs provide information about the latest run, as follows:

- **General Log**. This log includes all information data, errors, and warnings that occurred during the run.
- **Communication Log**. This log contains the detailed communication between the Discovery Analyzer and the remote destination (including its response). After the execution, the log can be saved as a record file.
- **Results Log**. Displays a list of discovered CIs. The appearance time of each CI depends on the design of the patterns and scripts.

You can save all logs together or each log separately. When you save all the logs, they are saved together under one name.

If you replay a record file, the same data is displayed in the communication log, the only difference being the time of execution.

Implement a Pattern

A DDM task has the aim of accessing remote (or local) systems, modeling extracted data as CIs, and saving the CIs to the CMDB. The task consists of the following steps:

1. **DDM pattern.**

   You configure a pattern file that holds the context, parameters, and result types for DDM by selecting the scripts that are to be part of the pattern. For details, see the following section.

2. **DDM job.**

   You configure a job with scheduling information and a trigger TQL. For details, see "Step 2: Assign a Job to the Pattern" on page 326.

3. **DDM code.**

   You can edit the Jython or Java code that is contained in the pattern files and that refers to the DDM Framework. For details, see "Step 3: Create Jython Code" on page 328.

   To write new DDM content, you create each of the above components, each one of which is automatically bound to the component in the previous step. For example, once you create a job and select the relevant pattern, the pattern file binds to the job.

Step 1: Create a Discovery and Dependency Mapping Pattern

A DDM pattern can be considered as the definition of a function. This function defines an input definition, runs logic on the input, defines the output, and provides a result.

Each DDM pattern specifies input and output: Both input and output are Trigger CIs that are specifically defined in the pattern. The DDM pattern extracts data from the input Trigger CI and passes this data as parameters to the DDM code. (Data from related CIs is sometimes passed to the code too. For details, see "Related CIs Window" on page 177.) A pattern's DDM code is generic, apart from these specific input Trigger CI parameters that are passed to the code.

For details on input components, see "Trigger CITs, Trigger CIs, Input TQLs, and Trigger TQLs" on page 58.

This section includes the following topics:

- "Define Pattern Input (Trigger CIT and Input TQL)" on page 318
- "Define Pattern Output" on page 323
- "Override Pattern Parameters" on page 324

Define Pattern Input (Trigger CIT and Input TQL)

You use the Trigger CIT and Input Topology Query Language (TQL) components to define specific CIs as pattern input:

- The Trigger CIT defines which CIT is used as the input for the pattern. For example, for a pattern that is going to discover IPs, the input CIT is Network.
- The Input TQL is a regular, editable TQL that defines the query against the CMDB. The Input TQL defines additional constraints on the CIT (for example, if the task requires a hostID or application_ip attribute), and can define more CI data, if needed by the pattern.

  If the pattern requires additional information from the CIs that are related to the Trigger CI, you can add additional nodes to the input TQL. For details, see "Example of Input TQL Definition" on page 319 and "Add Nodes and Relationships to a TQL Query" in *Model Management*.
- The Trigger CI data contains all the required information on the Trigger CI as well as information from the other nodes in the Input TQL, if they are defined. DDM uses variables to retrieve data from the CIs. When the task is downloaded to the Probe, the Trigger CI data variables are replaced with actual values that exist on the attributes for real CI instances.

Example of Trigger CIT Definition

In this example, a Trigger CIT defines that IP CIs are permitted in the pattern.

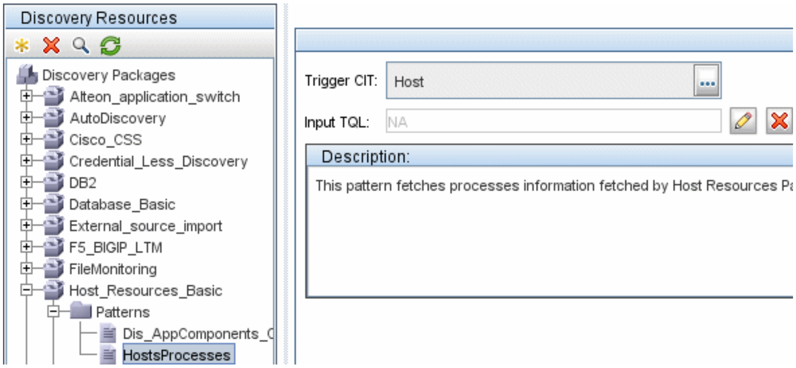
1. Access **Discovery** > **Manage Discovery Resources** > **Pattern Signature**. Select the HostProcesses pattern (**Discovery Packages** > **Host_Resources_Basic** > **Patterns** > **HostProcesses**).

2 Locate the Trigger CIT box. For details, see “Triggered CI Data Pane” on page 269.

3 Click the button to open the Choose Discovered Class dialog box. For details, see “Choose Discovered Class Dialog Box” on page 241.

4 Select the CIT.

In this example, the IP CI (Host) is permitted in the pattern:



Trigger CIT: Host

Input TQL: NA

Description:

This pattern fetches processes information fetched by Host Resources Pa

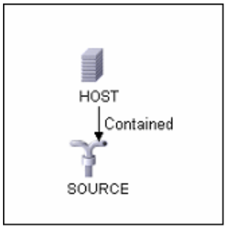
Example of Input TQL Definition

In this example, the Input TQL defines that the IP CI (configured in the previous example as the Trigger CIT) must be connected to a Host CI.

1 Access **Discovery** > **Manage Discovery Resources** > **Pattern Signature**. Locate the Input TQL box. Click the **Edit** button to open the Input TQL Editor. For details, see “Input TQL Editor Window” on page 251.

2 In the Input TQL Editor, name the Trigger CI node **SOURCE**: right-click the node and choose **Node Properties**. In the **Element Name** box, change the name to **SOURCE**.

3 Add a Host CI and a Contains relationship to the IP CI. For details on working with the Input TQL Editor, see "Input TQL Editor Window" on page 251.



The **IP** CI is connected to a **HOST** CI. The input TQL consists of two nodes, **HOST** and **IP**, with a link between them. The **IP** CI is named **SOURCE**.

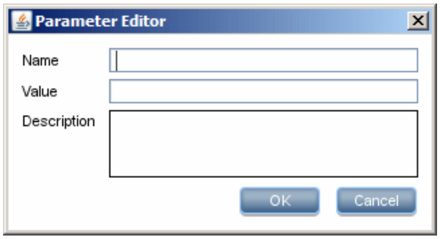
Example of Adding Variables to the Input TQL

In this example, you add DIRECTORY and CONFIGURATION_FILE variables to the Input TQL created in the previous example. These variables help to define what must be discovered, in this case, to find the configuration files residing on the hosts that are linked to the IPs you need to discover.

1 Display the Input TQL created in the previous example.

2 Access **Discovery** > **Manage Discovery Resources** > **Pattern Signature**. Locate the Triggered CI Data pane. For details, see "Triggered CI Data Pane" on page 269.

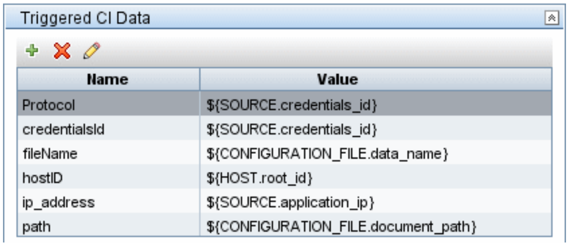
3 Add variables to the Input TQL. For details, see the Value field in the "Triggered CI Data Pane" on page 269.



Example of Replacing Variables with Actual Data

In this example, variables replace the IP CI data with actual values that exist on real IP CI instances in your system.

The Triggered CI data for the **IP** CI includes a fileName variable. This variable enables the replacement of the CONFIGURATION_FILE node in the Input TQL with the actual values of the configuration file located on a host:



Triggered CI Data

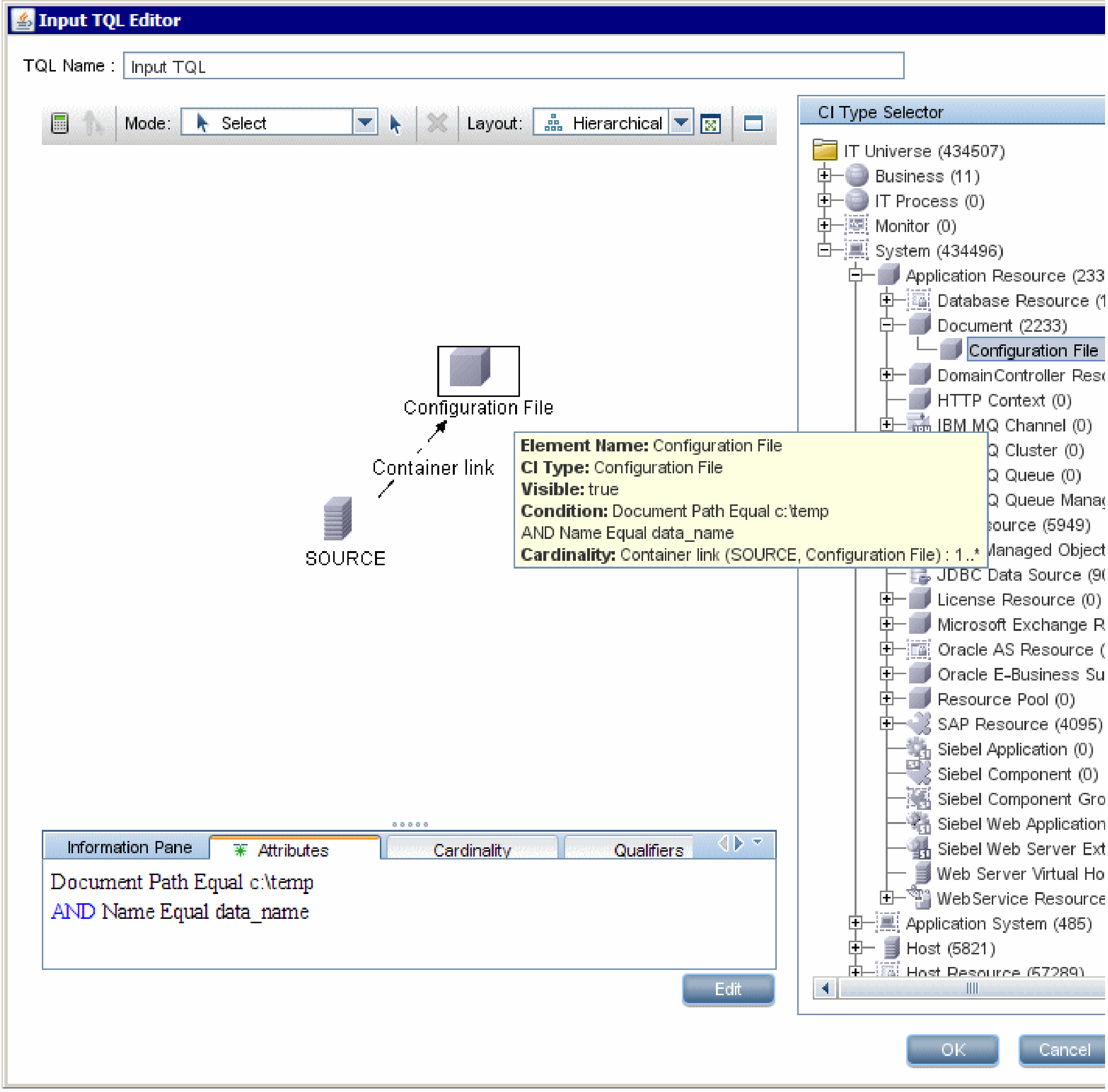
Name	Value
Protocol	${SOURCE.credentials_id}
credentialsId	${SOURCE.credentials_id}
fileName	${CONFIGURATION_FILE.data_name}
hostID	${HOST.root_id}
ip_address	${SOURCE.application_ip}
path	${CONFIGURATION_FILE.document_path}

The Trigger CI data is uploaded to the Probe with all variables replaced by actual values. The pattern script includes a command to use the DDM Framework to retrieve the actual values of the defined variables:

```
Framework.getTriggerCIData ('ip_address')
```

Note:

- The fileName and path variables use the data_name and document_path attributes from the Configuration File node (defined in the Input TQL – see previous example).



Input TQL Editor

TQL Name : Input TQL

Mode: Select

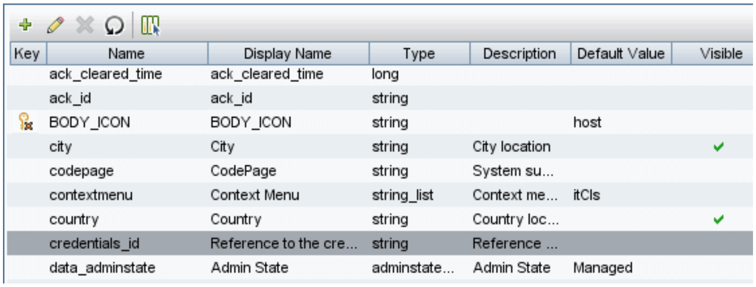
Layout: Hierarchical

Element Name: Configuration File
CI Type: Configuration File
Visible: true
Condition: Document Path Equal c:\temp
AND Name Equal data_name
Cardinality: Container link (SOURCE, Configuration File) : 1..*

Information Pane | Attributes | Cardinality | Qualifiers

Document Path Equal c:\temp
AND Name Equal data_name

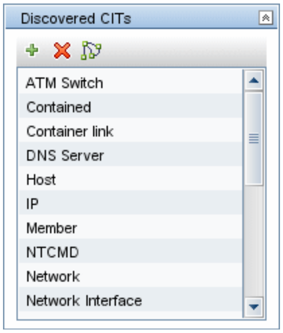
➤ The Protocol, credentialsId, and ip_address variables use the root_class, credentials_id, and application_ip attributes:



Key	Name	Display Name	Type	Description	Default Value	Visible
	ack_cleared_time	ack_cleared_time	long			
	ack_id	ack_id	string			
	BODY_ICON	BODY_ICON	string		host	
	city	City	string	City location		✓
	codepage	CodePage	string	System su...		
	contextmenu	Context Menu	string_list	Context me...	itCls	
	country	Country	string	Country loc...		✓
	credentials_id	Reference to the cre...	string	Reference ...		
	data_adminstate	Admin State	adminstate...	Admin State	Managed	

Define Pattern Output

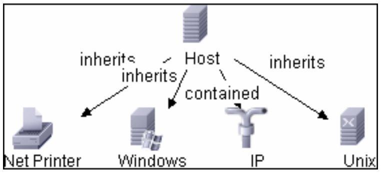
The output of the pattern is a list of discovered CIs (**Discovery** > **Manage Discovery Resources** > **Pattern Signature tab**) and the links between them:



Discovered CITs

- ATM Switch
- Contained
- Container link
- DNS Server
- Host
- IP
- Member
- NTCMD
- Network
- Network Interface

You can also view the CITs as a topology map, that is, the components and the way in which they are linked together (click the **View Discovered CITs as Map** button):



The discovered CIs are returned by the DDM code (that is, the Jython script) in the format of UCMDB's ObjectStateHolderVector. For details, see "Results Generation by the Jython Script" on page 334.

Example of Pattern Output

In this example, you define which CITs are to be part of the IP CI output.

1. Access **Discovery** > **Manage Discovery Resources**.
2. In the Discovery Resources pane, select **Network** > **Pattern** > **NSLOOKUP_on_Probe**.
3. In the Pattern Signature tab, locate the Discovered CITs pane.
4. The CITs that are to be part of the pattern output are listed. Add CITs to, or remove from, the list. For details, see "Discovered CITs Pane" on page 266.

Override Pattern Parameters

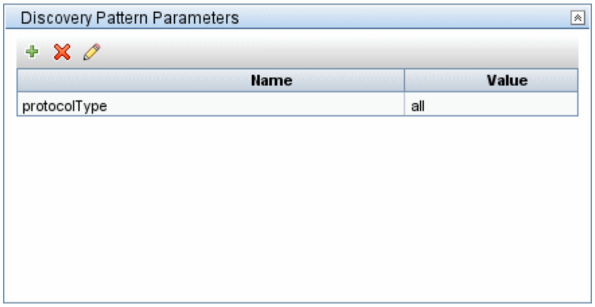
To configure a pattern for more than one job, you can override pattern parameters. For example, the pattern SQL_NET_Dis_Connection is used by both the MSSQL Connection by SQL and the Oracle Connection by SQL jobs.

Example of Overriding a Pattern Parameter

This example illustrates overriding a pattern parameter so that one pattern can be used to discover both Microsoft SQL Server and Oracle databases.

1. Access **Discovery** > **Manage Discovery Resources**.
2. In the Discovery Resources pane, select **Database Basic** > **Pattern** > **SQL_NET_Dis_Connection**.

3 In the Pattern Signature tab, locate the **Discovery Pattern Parameters** pane. The protocolType parameter has a value of **all**:

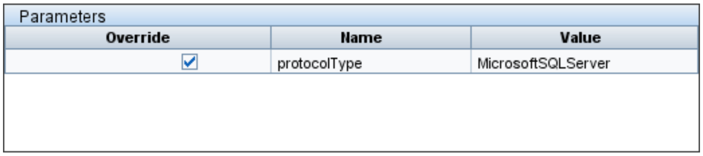


Discovery Pattern Parameters

Name	Value
protocolType	all

4 Right-click the **SQL_NET_Dis_Connection** pattern and choose **Go to Discovery Job** > **MSSQL Connection by SQL**.

5 Display the Properties tab. Locate the Parameters pane:



Parameters

Override	Name	Value
☑	protocolType	MicrosoftSQLServer

The all value is overwritten with the MicrosoftSQLServer value.

Note: The Oracle Connection by SQL job includes the same parameter but the value is overwritten with an oracle value.

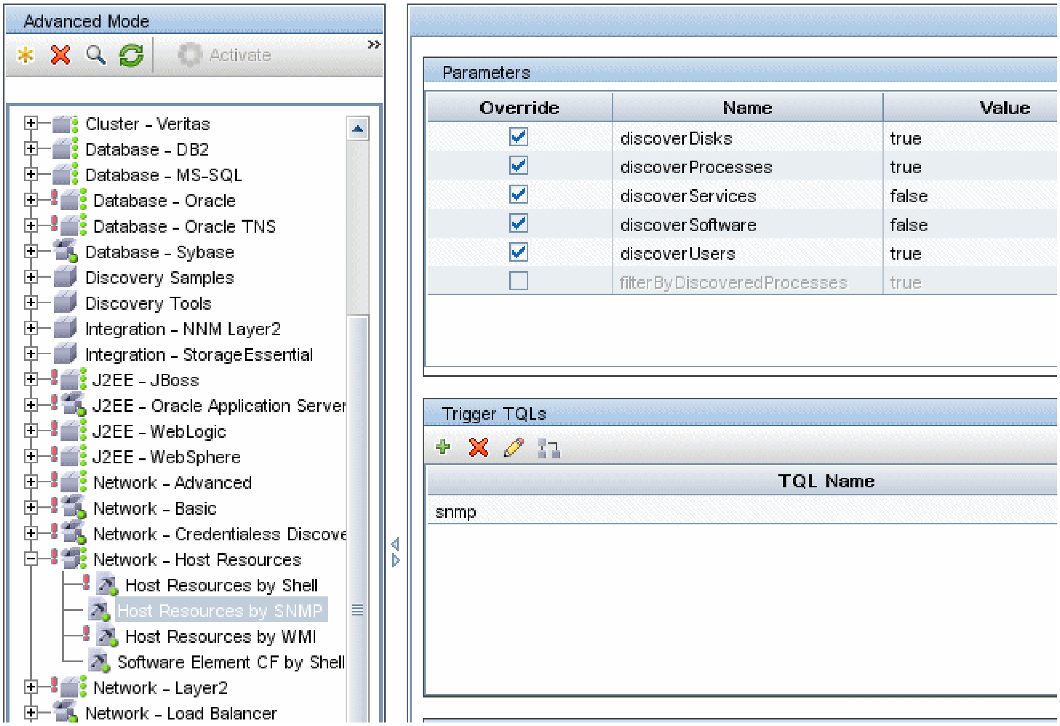
For details on adding, deleting, or editing parameters, see “Discovery Pattern Parameters Pane” on page 266.

DDM begins looking for Microsoft SQL Server instances according to this parameter.

Step 2: Assign a Job to the Pattern

Each pattern has one or more associated jobs that define the execution policy. Jobs enable scheduling the same pattern differently over different set of Triggered CIs and also enable supplying different parameters for each set.

The jobs appear in the Discovery Modules tree, and this is the entity that the user activates.



Cluster - Veritas
Database - DB2
Database - MS-SQL
Database - Oracle
Database - Oracle TNS
Database - Sybase
Discovery Samples
Discovery Tools
Integration - NNM Layer2
Integration - StorageEssential
J2EE - JBoss
J2EE - Oracle Application Server
J2EE - WebLogic
J2EE - WebSphere
Network - Advanced
Network - Basic
Network - Credentialess Discove
Network - Host Resources
Host Resources by Shell
Host Resources by SNMP
Host Resources by WMI
Software Element CF by Shell
Network - Layer2
Network - Load Balancer

Override	Name	Value
☑	discoverDisks	true
☑	discoverProcesses	true
☑	discoverServices	false
☑	discoverSoftware	false
☑	discoverUsers	true
□	filterByDiscoveredProcesses	true

TQL Name
snmp

Trigger TQL

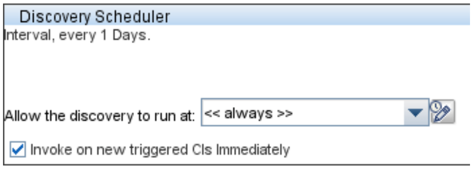
Each job is associated with Trigger TQLs. These Trigger TQLs publish results that are used as Input Trigger CIs for the pattern of this job.

A Trigger TQL can add constraints to an Input TQL. For example, if an input TQL's results are IPs connected to SNMP, a trigger TQL's results can be IPs connected to SNMP within the range 195.0.0.0-195.0.0.10.

Note: A trigger TQL must refer to the same objects that the input TQL refers to. For example, if an input TQL queries for IPs running SNMP, you cannot define a trigger TQL (for the same job) to query for IPs connected to a host, because some of the IPs may not be connected to an SNMP object, as required by the input TQL.

Scheduling

The scheduling information for the Probe specifies when to run the code on Trigger CIs. If the **Invoke on new triggered CIs Immediately** check box is selected, the code also runs once on each Trigger CI when it reaches the Probe, regardless of future schedule settings.



Discovery Scheduler

Interval, every 1 Days.

Allow the discovery to run at: << always >>

☑ Invoke on new triggered CIs Immediately

For each schedule occurrence for each job, the Probe runs the DDM code against all Trigger CIs accumulated for that job. For details, see “Schedule Modules to Run” on page 67 and “Discovery Scheduler Dialog Box” on page 149.

Parameters

When configuring a job you can override the pattern parameters. For details, see “Override Pattern Parameters” on page 324.

Step 3: Create Jython Code

DDM code is mostly written in Jython. "Jython is an implementation of the high-level, dynamic, object-oriented language Python written in 100% pure Java, and seamlessly integrated with the Java platform. It thus enables you to run Python on any Java platform." (Quoted from the Jython Web site: http://www.jython.org/Project/index.html.)

The following section describes the actual writing of Jython code inside the DDM Framework. This section specifically addresses those contact points between the Jython script and the Framework that it calls, and also describes the Jython libraries and utilities that should be used whenever possible.

Note:

- Scripts written for DDM should be compatible with Jython version 2.1.
- For full documentation on the available APIs, see the *HP Discovery and Dependency Mapping API Reference*.

This section includes the following topics:

- "Execution of the Code" on page 329
- "Modifying Out of the Box Scripts" on page 329
- "Structure of the Jython File" on page 331
- "Results Generation by the Jython Script" on page 334
- "The Framework Instance" on page 336
- "Finding the Correct Credentials (for Connection Patterns)" on page 340
- "Handling Exceptions from Java" on page 342
- "Jython Libraries and Utilities" on page 362
- "Using External Java JAR Files Within Jython" on page 315

Execution of the Code

After a job is activated, a task with all the required information is downloaded to the Probe.

The Probe starts running the DDM code using the information specified in the task.

The Jython code flow starts running from a main entry in the script, executes code to discover CIs, and provides results of a vector of discovered CIs.

Modifying Out of the Box Scripts

When making out of the box script modifications, make only minimal changes to the script and place any necessary methods in an external script. You can track changes more efficiently and, when moving to a newer HP Business Availability Center version, your code is not overwritten.

For example, the following single line of code in an out of the box script calls a method that calculates a Web server name in an application-specific way:

```
    serverName = iplanet_cspecific.PlugInProcessing(serverName, transportHN,
mam_utils)
```

The more complex logic that decides how to calculate this name is contained in an external script:

```
# implement customer specific processing for 'servername' attribute of httpplugin

#

def PlugInProcessing(servername, transportHN, mam_utils_handle):

        # support application-specific HTTP plug-in naming

        if servername == "appsrv_instance":

                      # servername is supposed to match up with the j2ee server name,
however some groups do strange things with their

                      # iPlanet plug-in files. this is the best work-around we could find. this join
can't be done with IP address:port

                      # because multiple apps on a web server share the same IP:port for
multiple websphere applications

                      logger.debug('httpcontext_webapplicationserver attribute has been
changed from [' + servername + '] to [' + transportHN[:5] + '] to facilitate websphere
enrichment')

                      servername = transportHN[:5]

        return servername
```

Save the external script in the External Resources folder. For details, see "Discovery Resources Pane" on page 246. If you add this script to a package, you can use this script for other jobs, too. For details on working with Package Manager, see "Package Manager" in *Model Management*.

During upgrade, the change you make to the single line of code is overwritten by the new version of the out of the box script, so you will need to replace the line. However, the external script is not overwritten.

Structure of the Jython File

The Jython file is composed of three parts in a specific order:

1. Imports
2. Functions definitions (optional)
3. Main Function - DiscoveryMain

The following is an example of a Jython script:

```
#   imports section
from appilog.common.system.types import ObjectStateHolder
from appilog.common.system.types.vectors import ObjectStateHolderVector

# Function definition
def foo:
 # do something

#   Main Function
def DiscoveryMain(Framework):
 OSHVResult = ObjectStateHolderVector()

 ## Write implementation to return new result CIs here...

 return OSHVResult
```

Imports

Jython classes are spread across hierarchical namespaces. In version 7.0 or later, unlike in previous versions, there are no implicit imports, and so every class you use must be imported explicitly. (This change was made for performance reasons and to enable an easier understanding of the Jython script by not hiding necessary details.)

➤ To import a Jython script:

```
import logger
```

➤ To import a Java class:

```
from appilog.collectors.clients import ClientsConsts
```

Main Function – DiscoveryMain

Each Jython runable script file contains a main function: DiscoveryMain.

The DiscoveryMain function is the main entry into the script; it is the first function that runs. The main function may call other functions that are defined in the scripts:

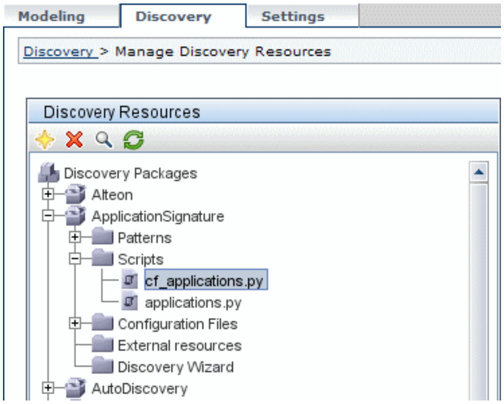
```
def DiscoveryMain(Framework):
```

The Framework argument must be specified in the main function definition. This argument is used by the main function to retrieve information that is required to run the scripts (such as information on the Trigger CI and parameters) and can also be used to report on errors that occur during the script run.

You can create a Jython script without any main method. Such scripts are used as library scripts that are called from other scripts.

Functions Definition

Each script can contain additional functions that are called from the main code. Each such function can call another function, which either exists in the current script or in another script (use the import statement). Note that to use another script, you must add it to the Scripts section of the package:



Example of a Function Calling Another Function

In the following example, the main code calls the doQueryOSUsers(..) method which calls an internal method doOSUserOSH(..):

```
def doOSUserOSH(name):
 sw_obj = ObjectStateHolder('winosuser')

 sw_obj.setAttribute('data_name', name)
 # return the object
 return sw_obj

def doQueryOSUsers(client, OSHVResult):
 _hostObj = modeling.createHostOSH(client.getIpAddress())
 data_name_mib = '1.3.6.1.4.1.77.1.2.25.1.1,1.3.6.1.4.1.77.1.2.25.1.2,string'
 resultSet = client.executeQuery(data_name_mib)
 while resultSet.next():
       UserName = resultSet.getString(2)
       ########## send object ##############
       OSUserOSH = doOSUserOSH(UserName)
       OSUserOSH.setContainer(_hostObj)
       OSHVResult.add(OSUserOSH)

def DiscoveryMain(Framework):
 OSHVResult = ObjectStateHolderVector()
 try:
       client =
Framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME).createClient()
 except:
       Framework.reportError('Connection failed')
 else:
       doQueryOSUsers(client, OSHVResult)
       client.close()
 return OSHVResult
```

If this script is a global library that is relevant to many patterns, you can add it to the list of scripts in the jythonGlobalLibs.xml configuration file, instead of adding it to each pattern (**Discovery** > **Manage Discovery Resources** > **Discovery Packages** > **AutoDiscovery** > **Configuration Files**).

Results Generation by the Jython Script

Each Jython script runs on a specific Trigger CI, and ends with results that are returned by the return value of the DiscoveryMain function.

The script result is actually a group of CIs and links that are to be inserted or updated in the CMDB. The script returns this group of CIs and links in the format of ObjectStateHolderVector.

The ObjectStateHolder class is a way to represent an object or link defined in the CMDB. The ObjectStateHolder object contains the CIT name and a list of attributes and their values. The ObjectStateHolderVector is a vector of ObjectStateHolder instances.

The ObjectStateHolder Syntax

This section explains how to build the DDM results into a CMDB model.

Example of Setting Attributes on the CIs

The ObjectStateHolder class describes the DDM result graph. Each CI and link (relationship) is placed inside an instance of the ObjectStateHolder class as in the following Jython code sample:

```
# siebel application server
1 appServerOSH = ObjectStateHolder('siebelappserver' )
2 appServerOSH.setStringAttribute('data_name', sblsvrName)
3 appServerOSH.setStringAttribute ('application_ip', ip)
4 appServerOSH.setContainer(appServerHostOSH)
```

- Line 1 creates a CI of type **siebelappserver**.
- Line 2 creates an attribute called **data_name** with a value of **sblsvrName** which is a Jython variable set with the value discovered for the server name.
- Line 3 sets a non-key attribute that is updated in the CMDB.
- Line 4 is the building of containment (the result is a graph). It specifies that this application server is contained inside a host (another ObjectStateHolder class in the scope).

Note: Each CI being reported by the Jython script must include values for all the key attributes of the CI's CI Type.

Example of Relationships (Links)

The following link example explains how the graph is represented:

```
1 linkOSH = ObjectStateHolder('route')
2 linkOSH.setAttribute('link_end1', gatewayOSH)
3 linkOSH.setAttribute('link_end2', appServerOSH)
```

- Line 1 creates the link (that is also of the ObjectStateHolder class. The only difference is that route is a link CI Type).
- Lines 2 and 3 specify the nodes at the end of each link. This is done using the **end1** and **end2** attributes of the link which must be specified (because they are the minimal key attributes of each link). The attribute values are ObjectStateHolder instances. For details on End 1 and End 2, see "Link" on page 242.

Important: A link is directional. You should verify that End 1 and End 2 nodes correspond to valid CITs at each end. If the nodes are not valid, the result object fails DDM validation and is not reported correctly. For details, see "CI Type Relationships" in *Model Management*.

Example of Vector (Gathering CIs)

After creating objects with attributes, and links with objects at their ends, you must now group them together. You do this by adding them to an ObjectStateHolderVector instance, as follows:

```
oshvMyResult = ObjectStateHolderVector()
oshvMyResult.add(appServerOSH)
oshvMyResult.add(linkOSH)
```

For details on reporting this composite result to the Framework so it can be sent to the CMDB server, see the sendObjects method.

Once the DDM result graph is assembled in an ObjectStateHolderVector instance, it must be returned to the DDM Framework to be inserted into the CMDB. This is done by returning the ObjectStateHolderVector instance as the result of the DiscoveryMain() function.

Note: For details on creating **OSH** for common CITs, see **modeling.py** in “Jython Libraries and Utilities” on page 362.

The Framework Instance

The Framework instance is the only argument that is supplied in the main function in the Jython script. This is an interface that can be used to retrieve information required to run the script (for example, information on trigger CIs and pattern parameters), and is also used to report on errors that occur during the script run. For details, see “HP Discovery and Dependency Mapping API Reference” on page 315.

This section describes the most important Framework usages:

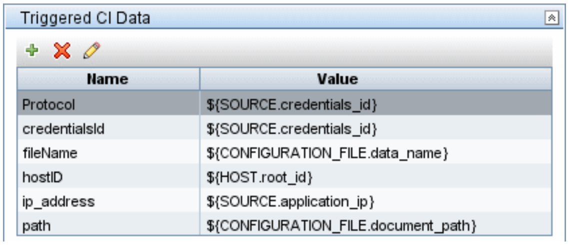
- “Framework.getTriggerCIData(String attributeName)” on page 336
- “Framework.createClient(credentialsId, props)” on page 337
- “Framework.getParameter (String parameterName)” on page 339
- “Framework.reportError(String message) and Framework.reportWarning(String message)” on page 339

Framework.getTriggerCIData(String attributeName)

This API provides the intermediate step between the Trigger CI data defined in the pattern and the script.

Example of Retrieving Credential Information

You request the following Trigger CI data information:



Triggered CI Data

Name	Value
Protocol	${SOURCE.credentials_id}
credentialsId	${SOURCE.credentials_id}
fileName	${CONFIGURATION_FILE.data_name}
hostID	${HOST.root_id}
ip_address	${SOURCE.application_ip}
path	${CONFIGURATION_FILE.document_path}

To retrieve the credential information from the task, use this API:

```
credId = Framework.getTriggerCIData('credentialsId')
```

Framework.createClient(credentialsId, props)

You make a connection to a remote machine by creating a client object and executing commands on that client. To create a client, retrieve the ClientFactory class. The getClientFactory() method receives the type of the requested client protocol. The protocol constants are defined in the ClientsConsts class. For details on credentials and supported protocols, see “Domain Credential References” on page 204.

Example of Creating a Client Instance for the Credentials ID

To create a Client instance for the credentials ID:

```
properties = Properties()
codePage = Framework.getCodePage()
properties.put( BaseAgent.ENCODING, codePage)
client = Framework.createClient(credentailsID ,properties)
```

You can now use the Client instance to connect to the relevant machine or application.

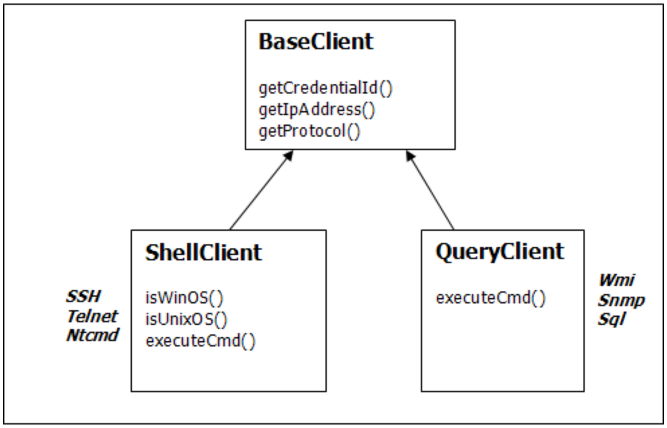
Example of Creating a WMI Client and Running a WMI Query

To create a WMI client and run a WMI query using the client:

```
wmiClient = Framework.createClient(credential)
resultSet = wmiClient. executeQuery("SELECT TotalPhysicalMemory
                                     FROMWin32_LogicalMemoryConfiguration")
```

Note: To make the createClient() API work, add the following parameter to the Trigger CI data parameters: **credentialsId = ${SOURCE.credentials_id}** in the Triggered CI Data pane. Or you can manually add the credentials ID when calling the function:
wmiClient = clientFactory().createClient(credentials_id).

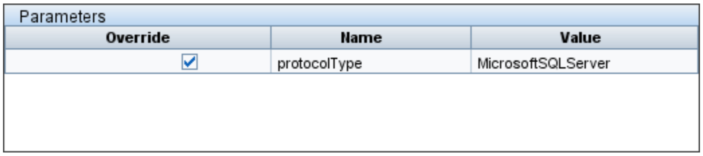
The following diagram illustrates the hierarchy of the clients, with their commonly-supported APIs:



For details on the clients and their supported APIs, see BaseClient, ShellClient, and QueryClient in the *HP Discovery and Dependency Mapping API Reference*.

Framework.getParameter (String parameterName)

In addition to retrieving information on the Trigger CI, you often need to retrieve a pattern parameter value. For example:



Parameters

Override	Name	Value
☑	protocolType	MicrosoftSQLServer

Example of Retrieving the Value of the protocolType Parameter

To retrieve the value of the protocolType parameter from the Jython script, use the following API:

```
protocolType = Framework.getParameterValue('protocolType')
```

Framework.reportError(String message) and Framework.reportWarning(String message)

Some errors (for example, connection failure, hardware problems, timeouts) can occur during a script run. When such errors are detected, Framework can report on the problem. The message that is reported reaches the server and is displayed for the user.

Example of a Report Error and Message

The following example illustrates the use of the reportError(<Error Msg>) API:

```
try:
        client = Framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME)
        createClient()
except:
        strException = str(sys.exc_info()[1]).strip()
        Framework. reportError ('Connection failed: %s' % strException)
```

You can use either one of the APIs—Framework.reportError(String message), Framework.reportWarning(String message)—to report on a problem. The difference between the two APIs is that when reporting an error, the Probe saves a communication log file with the entire session's parameters to the file system. In this way you are able to track the session and better understand the error.

Finding the Correct Credentials (for Connection Patterns)

A pattern trying to connect to a remote system needs to try all possible credentials. One of the parameters needed when creating a client (through ClientFactory) is the credentials ID. The connection script gains access to possible credential sets and tries them one by one using the clientFactory.getAvailableProtocols() method. When one credential set succeeds, the pattern reports a CI connection object on the host of this trigger CI (with the credentials ID that matches the IP) to the CMDB. Subsequent patterns can use this connection object CI directly to connect to the credential set (that is, the patterns do not have to try all possible credentials again).

The following example shows how to obtain all entries of the SNMP protocol. Note that here the IP is obtained from the Trigger CI data (# Get the Trigger CI data values).

The connection script requests all possible protocol credentials (# Go over all the protocol credentials) and tries them in a loop until one succeeds (resultVector). For details, see the **two-phase connect paradigm** entry in "Separating Patterns" on page 313.

```
import logger
from appilog.collectors.clients import ClientsConsts
from appilog.common.system.types.vectors import ObjectStateHolderVector

    def mainFunction(Framework):
resultVector = ObjectStateHolderVector()

    # Get the Trigger CI data values
    ip_address = Framework.getDestinationAttribute('ip_address')
    ip_domain  = Framework.getDestinationAttribute('ip_domain')

    # Create the client factory for SNMP
    clientFactory = framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME)
    protocols = clientFactory.getAvailableProtocols(ip_address, ip_domain)

    connected = 0
    # Go over all the protocol credentials
    for credentials_id in protocols:
        client = None
        try:
            # try to connect to the snmp agent
            client = clientFactory.createClient(credentials_id)

            // Query the agent
            ….

            # connection succeed
            connected = 1
        except:
            if client != None:
                client.close()
    if (not connected):
        logger.debug('Failed to connect using all credentials')
    else:
        // return the results as OSHV
        return resultVector
```

Handling Exceptions from Java

Some Java classes throw an exception upon failure. It is recommended to catch the exception and handle it, otherwise it causes the pattern to terminate unexpectedly.

When catching a known exception, in most cases you should print its stack trace to the log and issue a proper message to the UI, for example:

```
try:
	client = Framework.getClientFactory().createClient()
except Exception, msg:
	Framework.reportError('Connection failed')
	logger.debugException('Exception while connecting: %s' % (msg))
	return
```

If the exception is not fatal and the script can continue, you should omit the call for the reportError() method and enable the script to continue.

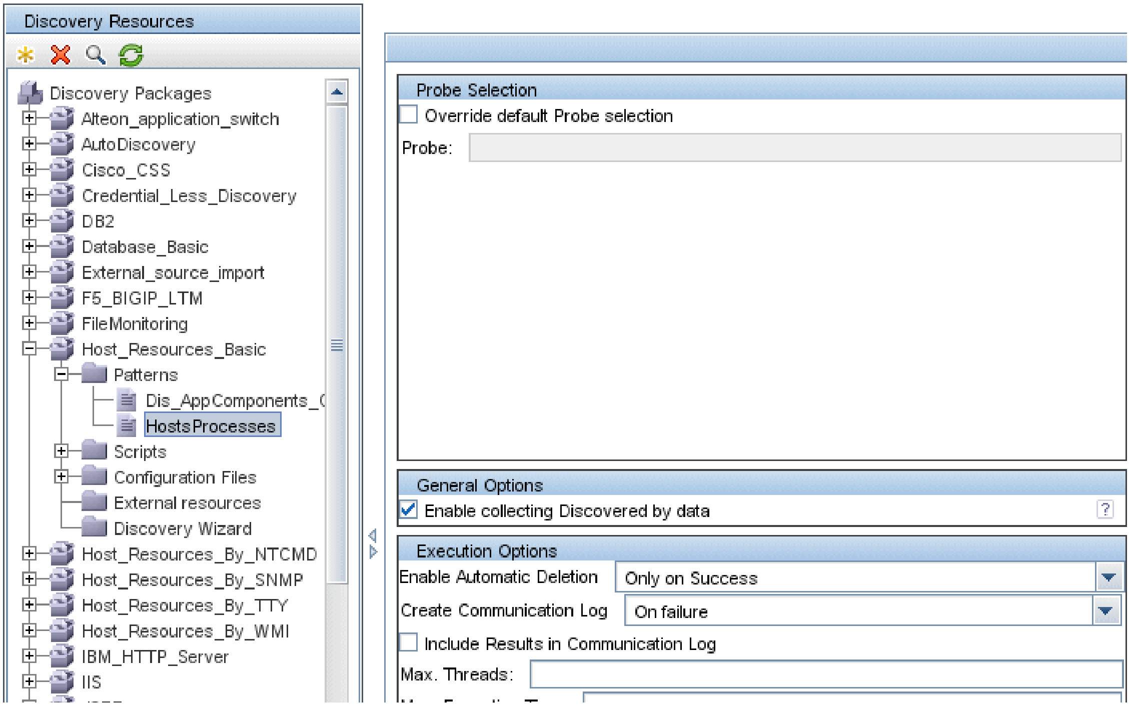
Record DDM Code

It can be very useful to record an entire execution, including all parameters, for example, when debugging and testing code. This task describes how to record an entire execution with all relevant variables. Furthermore, you can view extra debug information that is usually not printed to log files even at the debug level.

To record DDM code:

1 Access **Discovery** > **Run Discovery**. Right-click the job whose run must be logged and select **Edit pattern** to open the Manage Discovery Resources application.

2 Locate the **Execution Options** pane in the Pattern Management tab:



Discovery Resources

- Discovery Packages
  - Alteon_application_switch
  - AutoDiscovery
  - Cisco_CSS
  - Credential_Less_Discovery
  - DB2
  - Database_Basic
  - External_source_import
  - F5_BIGIP_LTM
  - FileMonitoring
  - Host_Resources_Basic
    - Patterns
      - Dis_AppComponents_(
      - HostsProcesses
    - Scripts
    - Configuration Files
    - External resources
    - Discovery Wizard
  - Host_Resources_By_NTCMD
  - Host_Resources_By_SNMP
  - Host_Resources_By_TTY
  - Host_Resources_By_WMI
  - IBM_HTTP_Server
  - IIS

Probe Selection

□ Override default Probe selection

Probe:

General Options

☑ Enable collecting Discovered by data

Execution Options

Enable Automatic Deletion: Only on Success

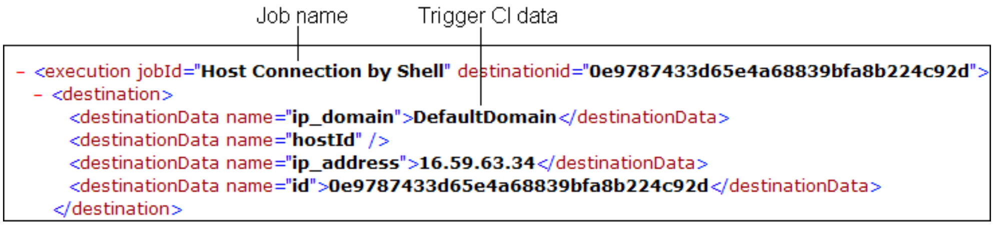
Create Communication Log: On failure

□ Include Results in Communication Log

Max. Threads:

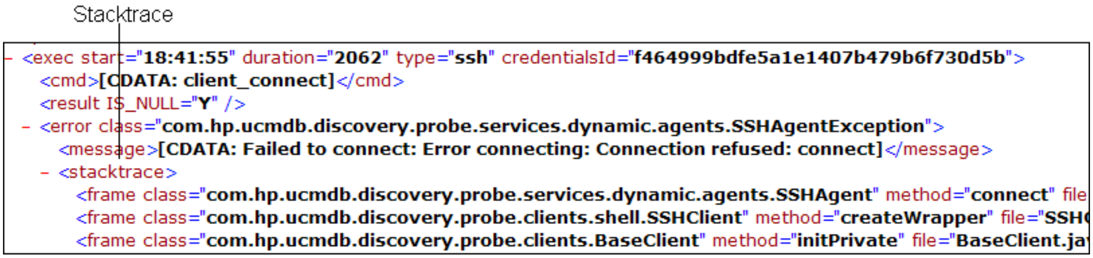
3 Change the **Create communication logs** box to **Always**. For details on setting logging options, see "Execution Options Pane" on page 260.

The following example is the XML log file that is created when the Host Connection by Shell job is run and the **Create communication logs** box is set to **Always** or **On Failure**:



```
- <execution jobId="Host Connection by Shell" destinationid="0e9787433d65e4a68839bfa8b224c92d">
  - <destination>
      <destinationData name="ip_domain">DefaultDomain</destinationData>
      <destinationData name="hostId" />
      <destinationData name="ip_address">16.59.63.34</destinationData>
      <destinationData name="id">0e9787433d65e4a68839bfa8b224c92d</destinationData>
    </destination>
```

The following example shows the message and stacktrace parameters:



Stacktrace

```
- <exec start="18:41:55" duration="2062" type="ssh" credentialsId="f464999bdfe5a1e1407b479b6f730d5b">
    <cmd>[CDATA: client_connect]</cmd>
    <result IS_NULL="Y" />
  - <error class="com.hp.ucmdb.discovery.probe.services.dynamic.agents.SSHAgentException">
      <message>[CDATA: Failed to connect: Error connecting: Connection refused: connect]</message>
    - <stacktrace>
        <frame class="com.hp.ucmdb.discovery.probe.services.dynamic.agents.SSHAgent" method="connect" file
        <frame class="com.hp.ucmdb.discovery.probe.clients.shell.SSHClient" method="createWrapper" file="SSHC
        <frame class="com.hp.ucmdb.discovery.probe.clients.BaseClient" method="initPrivate" file="BaseClient.ja
```

Configure Eclipse to Run Jython Scripts in Debug Mode

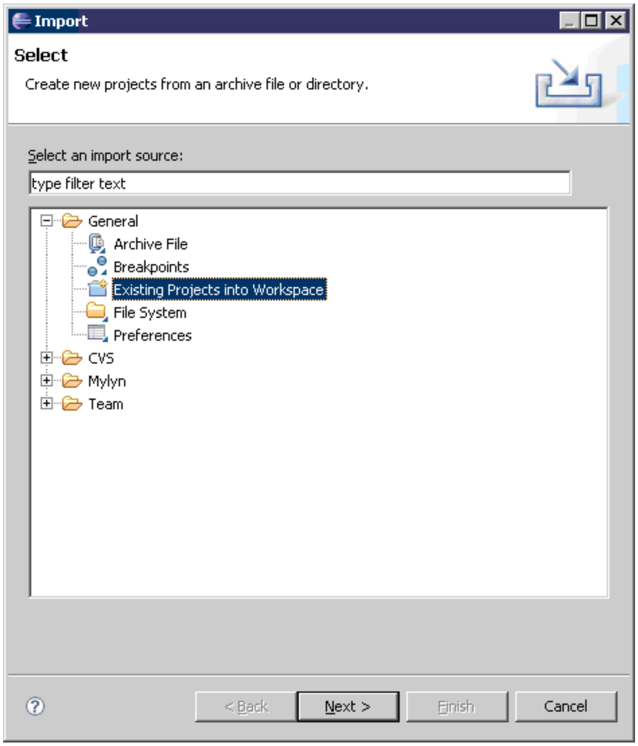
This task explains how to configure Eclipse so that you can run your Jython scripts in debug mode, thus enabling better visibility to job threads, trigger CIs, and results.

To configure Eclipse:

1 Prerequisites:

- Install Java 5.0 SE on your computer (if it is not already installed). The files are available on the Java Developers Web site http://www.java.sun.com.
- Install the latest Eclipse version on your computer. The application is available at www.eclipse.org.
- Install PyDev and PyDev Extensions, available at: **http://pydev.sourceforge.net/** (PyDev) and **http://www.fabioz.com/pydev/** (PyDev Extensions)
- The Discovery Probe must be installed on this computer.

2 Select **File** > **Import** > **General** > **Existing Projects into Workspace**:



Import

Select

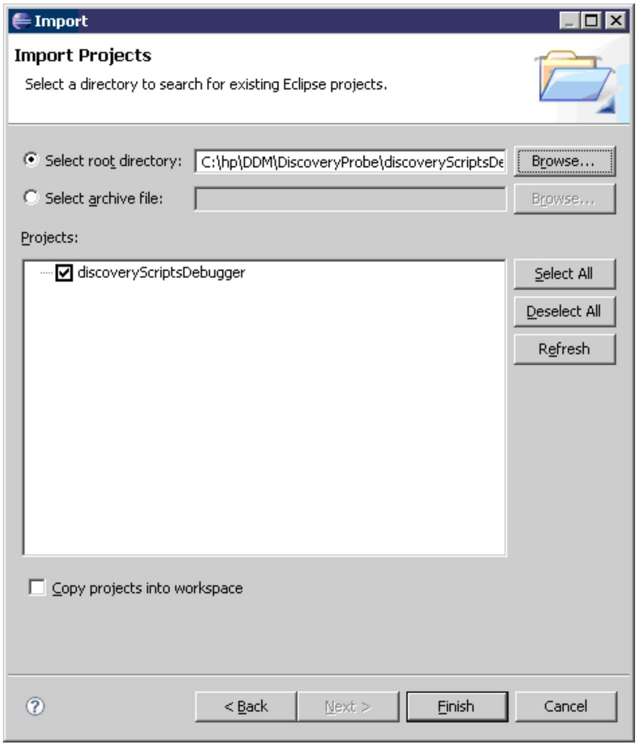
Create new projects from an archive file or directory.

Select an import source:

type filter text

3 Click **Next**. Click **Browse** to select the following directory: **C:\hp\DDM\DiscoveryProbe\discoveryScriptsDebugger**:

4 Verify that **discoveryScriptsDebugger** is selected in the Projects pane and click **Finish**:



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\hp\DDM\DiscoveryProbe\discoveryScriptsDe

○ Select archive file:

Projects:

☑ discoveryScriptsDebugger

□ Copy projects into workspace

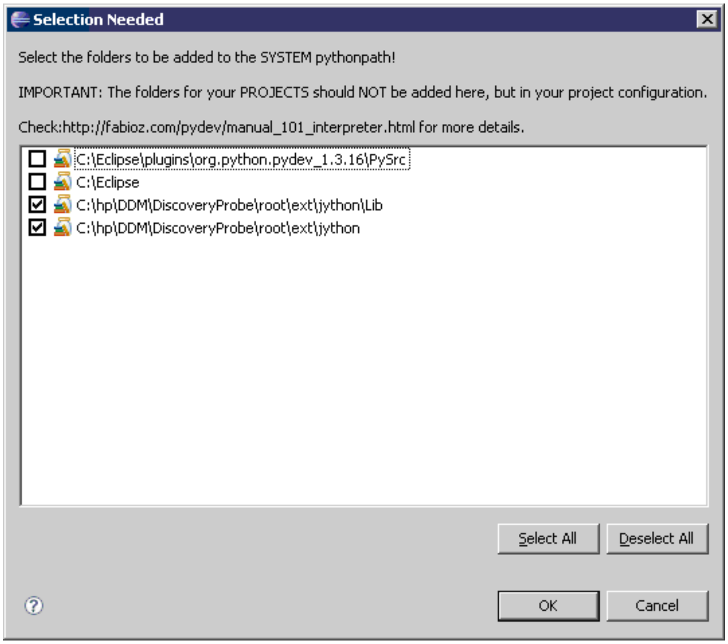
A discoveryScriptsDebugger project is added to the Eclipse workspace.

5 Select **Window** > **Preferences** to open the **Preferences** dialog box. Select **Pydev** > **Interpreter - Jython** > **New**.

6 Select the following directories:

➤ **C:\hp\DDM\DiscoveryProbe\root\ext\jython\**

➤ **C:\hp\DDM\DiscoveryProbe\root\ext\jython\lib**



Selection Needed

Select the folders to be added to the SYSTEM pythonpath!

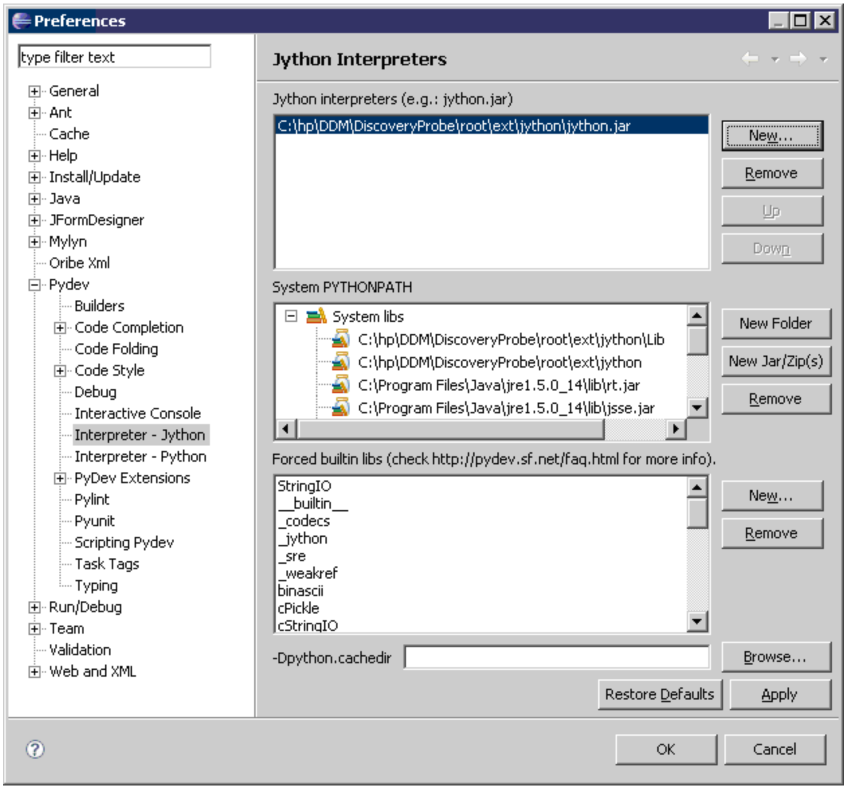
IMPORTANT: The folders for your PROJECTS should NOT be added here, but in your project configuration.

Check:http://fabioz.com/pydev/manual_101_interpreter.html for more details.

- □ C:\Eclipse\plugins\org.python.pydev_1.3.16\PySrc
- □ C:\Eclipse
- ☑ C:\hp\DDM\DiscoveryProbe\root\ext\jython\Lib
- ☑ C:\hp\DDM\DiscoveryProbe\root\ext\jython

Click **OK**.

7 Return to the **Preferences** dialog box. Verify that the jython.jar file appears in the Jython interpreters list:



Jython interpreters (e.g.: jython.jar)

C:\hp\DDM\DiscoveryProbe\root\ext\jython\jython.jar

System PYTHONPATH

- System libs
  - C:\hp\DDM\DiscoveryProbe\root\ext\jython\Lib
  - C:\hp\DDM\DiscoveryProbe\root\ext\jython
  - C:\Program Files\Java\jre1.5.0_14\lib\rt.jar
  - C:\Program Files\Java\jre1.5.0_14\lib\jsse.jar

Forced builtin libs (check http://pydev.sf.net/faq.html for more info).

```
StringIO
__builtin__
_codecs
_jython
_sre
_weakref
binascii
cPickle
cStringIO
```

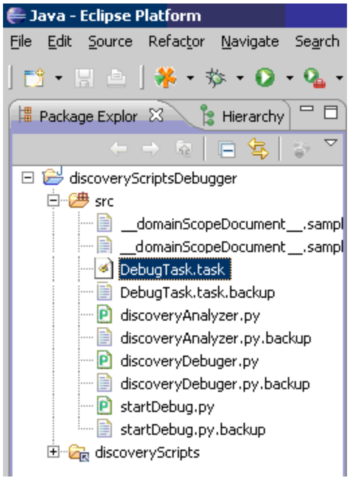
-Dpython.cachedir

Click **OK**.

8 Select **Window** > **Open Perspective** > **Other** to open the **Open Perspective** dialog box. Select **PyDev**.

9 You can configure the job to be debugged either in a text editor or by using a graphic utility. For details on the text editor configuration, go to the next step. For details on the graphic utility configuration, skip to step 13 on page 351.

10 Select the **DebugTask.task** file in the **discoveryScriptsDebugger\src** project:

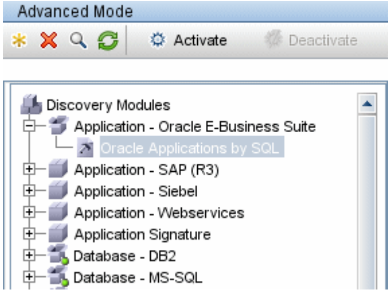


To edit this file in Eclipse, right-click **DebugTask.task** and select **Open With** > **Text Editor**.

The DebugTask.task file is a template you can use to debug your Jython scripts. It is located in the following directory: **C:\hp\DDM\DiscoveryProbe\discoveryScriptsDebugger\src**.

```
<taskInfo taskId="FOO Debug Task OOF">
 <jobId>Range IPs by ICMP</jobId>
 <destinationList>
      <destination>
         <destinationData name="domain">DefaultDomain</destinationData>
         <destinationData name="probeName">DefaultProbe</destinationData>
      </destination>
 </destinationList>
</taskInfo>
```

11 Change the jobid value to the name of the job to be debugged. For example, delete **Range IPs by ICMP** and enter **Oracle Applications by SQL**. You can copy the name from the job name in the Run Discovery page:



12 Add as many destinationData attributes as necessary (do not change the destinationList className). For example, to debug the **Oracle Applications by SQL** script, enter the following values:

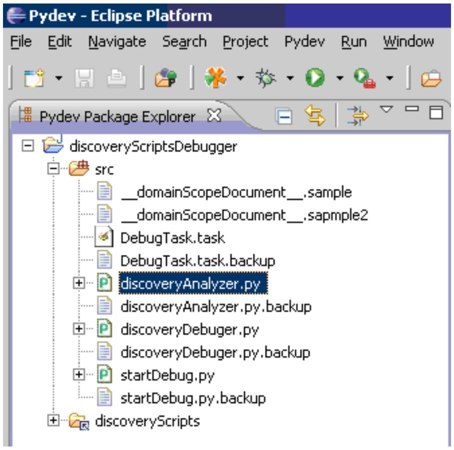
```
  <jobId>IIS Applications by NTCMD</jobId>
  <destinationList
className="com.hp.ucmdb.discovery.probe.tasks.BaseDestinationData">
      <destination>
          <destinationData name="fileToUpdate">discoveryJobs/Oracle Applications by
SQL.xml</destinationData>
          <destinationData name="fileUpdateTime">1216821680058</destinationData>
          <destinationData
name="id">bfd3d1ea8f7229e055c7004cf87ebd65</destinationData>
          <destinationData name="hostId" />
          <destinationData name="ip_address" />
          <destinationData name="ip_domain" />
      </destination>
  </destinationList>
```

You can verify the exact destinationData attributes for each job by consulting the following log file: **C:\hp\DDM\DiscoveryProbe\root\logs\probeGW-taskResults.log**.

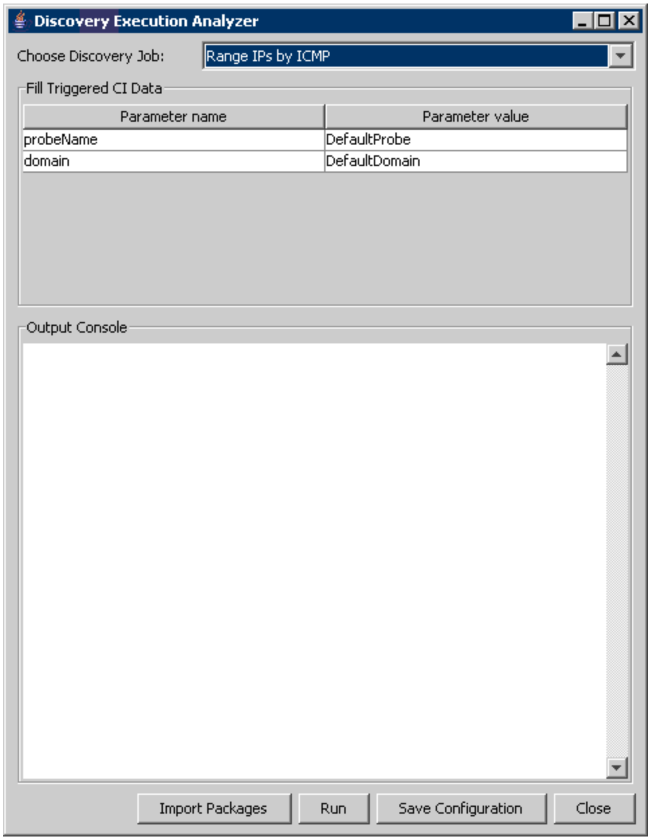
Note: Disregard UPDATE_SERVER_DATA jobs in the log file.

Skip to step 14 on page 352.

13 To use the graphic utility, select the **discoveryAnalyzer.py** file in the **discoveryScriptsDebugger****src** project. Right-click the file and choose **Run as** > **Jython run**.



The Discovery Execution Analyzer dialog box is displayed:



Discovery Execution Analyzer

Choose Discovery Job: Range IPs by ICMP

Fill Triggered CI Data

Parameter name	Parameter value
probeName	DefaultProbe
domain	DefaultDomain

Output Console

Choose the Discovery Job, fill in the parameter values and save the configuration.

14 Add a breakpoint in the Jython script to be debugged.

15 In the **discoveryScriptsDebugger** project, right-click **startDebug.py** and choose **Debug As** > **Jython Run**.

Troubleshooting and Limitations

The **pydev** plug-in including the Jython interpreter should be well configured. The recommended way is to select the Jython interpreter that is part of the DDM Probe installation: **Eclipse window** > **Preferences**. The workspace should be used that includes a **.pydevproject** defining the class path of **pydev**. It is selected automatically if the **pydev** is correctly installed.

Work with Discovery Analyzer

The following procedure explains how to work with Discovery Analyzer.

This section includes the following topics:

- "Prerequisites" on page 354
- "Access Discovery Analyzer" on page 354
- "Define a Task" on page 355
- "Define a New Task" on page 356
- "Retrieve a Record" on page 357
- "Open a Task File" on page 357
- "Import a Job from the Database" on page 357
- "Edit a Task" on page 357
- "Save the Task and Logs" on page 358
- "Run the Task" on page 358
- "Send a Task Result to the Server" on page 358
- "Import Settings" on page 359
- "Breakpoints" on page 359

1 Prerequisites

- The Probe must be installed. (The Discovery Analyzer is installed as part of the Probe installation process and shares resources with it.)
- The Probe does not need to be running while you are working with Discovery Analyzer.

  However, if the Probe has already run against a UCMDB server, all the required resources are already downloaded to the file system. If the Probe has not run, you can upload resources needed by Discovery Analyzer through the Settings menu. For details, see "Import Settings" on page 359.
- The UCMDB server does not need to be installed.

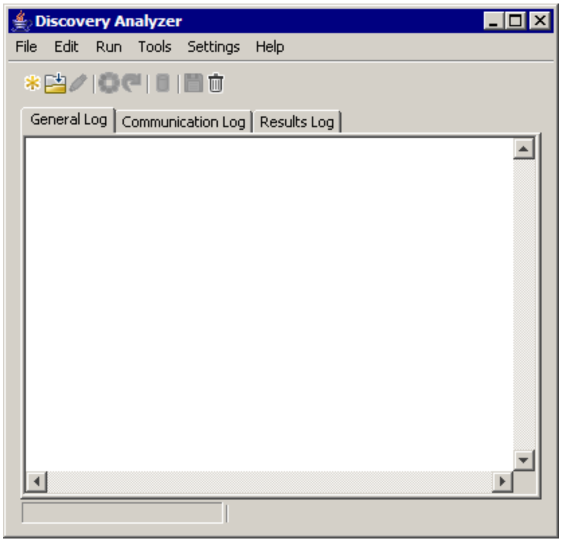
2 Access Discovery Analyzer

You access Discovery Analyzer either:

- When working with Eclipse. The recommended plug-in is **pydev**. Install PyDev and PyDev Extensions, available at: **http://pydev.sourceforge.net/** (PyDev) and **http://www.fabioz.com/pydev/** (PyDev Extensions).

  The Probe installation comes with a default Eclipse workspace located at **C:\hp\DDM\DiscoveryProbe\discoveryAnalyzerWorkspace**. This workspace includes a Jython script to start Discovery Analyzer (**startDiscoveryAnalyzerScript.py**) as well as a link to all DDM scripts. If you start the tool in this way, you can locate breakpoints within the Jython scripts for debugging purposes.
- Directly, by double-clicking the file in the following folder: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\discoveryAnalyzer.cmd**. For details, see the following section.

The Discovery Analyzer window opens:



3 Define a Task

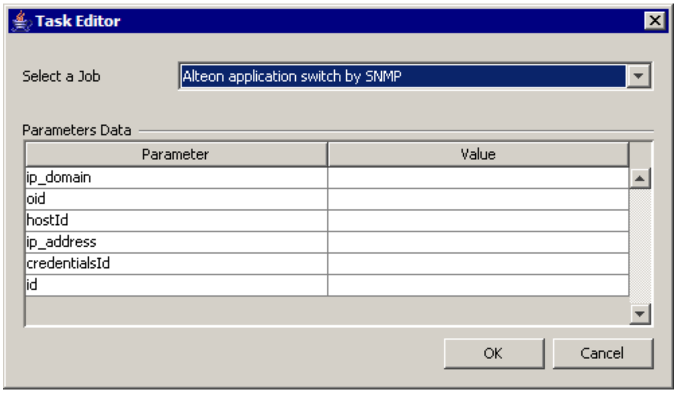
You define a task using one of the following methods:

- ➤ By defining a new task. For details, see “Define a New Task” on page 356.
- ➤ By importing a task from a record file. For details, see “Retrieve a Record” on page 357.
- ➤ By importing a saved task from a task file. For details, see “Open a Task File” on page 357.
- ➤ By retrieving a job from the Probe’s internal database. For details, see “Import a Job from the Database” on page 357.

4 Define a New Task

a Display the Task Editor: click the **New Task** button .

The Task Editor displays a list of jobs that currently exist in the file system. This list is updated each time the Probe receives tasks from the server, or packages are deployed manually from the Settings menu.



Select a Job: Alteon application switch by SNMP

Parameters Data

Parameter	Value
ip_domain	
oid	
hostId	
ip_address	
credentialsId	
id	

b Select a job.

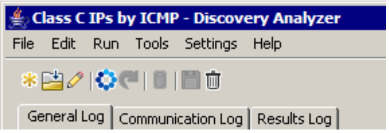
c Enter values for all parameters.

The parameters displayed here are DDM pattern parameters. They can be viewed in the Discovery Pattern Parameters pane in the Pattern Signature tab. For details, see “Discovery Pattern Parameters Pane” on page 266.

All fields are mandatory (unless a job’s script demands that the field be empty).

For parameters that require an ID or credentials ID input value, you can use randomly created IDs: right-click the value box and select **Generate random CMDB ID** or **Credential Chooser**.

The task is now active and the name of the open task is displayed in the title bar:



Class C IPs by ICMP - Discovery Analyzer

d Continue with the procedure for defining a task. For details, see “Save the Task and Logs” on page 358.

5 Retrieve a Record

You can define a task by opening a record file containing data regarding a specific execution. If a task is defined in this way, you can reproduce the specific execution by selecting the playback option. (If a task is replayed, responses are received from the data stored in the record file and not from the remote destination.)

Select **File** > **Open Record**. Browse to the folder where you saved the record. The record is now active and the name of the task is displayed in the title bar.

6 Open a Task File

You can define a task from a task file: Select **File** > **Open Task**.

7 Import a Job from the Database

You can retrieve a job from the Probe database on condition that the Probe has already run and has active tasks in its internal database. You can use the parameter values to define the task.

a Select **File** > **Import Task from Probe Database**.

b In the dialog box that opens, select the job to run and click **OK**.

c Continue with the procedure for defining a task. For details, see “Save the Task and Logs” on page 358.

8 Edit a Task

After a task is defined, the name of the task (or the file) is displayed in the title bar. Now the file can be edited.

a Select **Edit** > **Edit Task**.

b Make any changes to the task and click **OK**.

9 Save the Task and Logs

You can save task parameters: Select **File** > **Save Task**.

The following options are available only after a task is executed.

➤ Save a record of the task. You can save the task parameters and the results of the task run: Select **File** > **Save Record**.

➤ Save a log of the task: Select **File** > **Save General Log**.

➤ Save results: Select **File** > **Save Results**.

10 Run the Task

The next step in the procedure is to run the task you created.

a To execute the task only against a remote destination, click the **Run Task** button.

Discovery Analyzer executes the job and displays information in the three log files: **General**, **Communication**, and **Results**.

b You can save the log files, either together or separately: Select **File** > **Save General Log**, **Save Record**, **Save Results**, or **Save All Logs**. For details on the log files, see "Logs" on page 316.

c If a task is retrieved from a record file, the execution that is documented in this file can be reproduced by clicking the **Playback** button. The same Communication log is displayed, but the execution time is updated.

11 Send a Task Result to the Server

If a task's execution ends with results (that is, the Results Log tab displays a list of discovered CIs), you can send the results to the UCMDB server. This is useful if, for example, you were previously testing a script when the server was down.

Note: You can send results only to a UCMDB server that receives tasks from the Probe that is installed on the same machine as Discovery Analyzer.

---

12 Import Settings

If the Probe has not yet run, you can import files needed by Discovery Analyzer. Access the Settings menu and import **domainScopeDocument.xml** or **domainScopeDocument.bin**. If you import the **.bin** file, you must import **key.bin** too. You can deploy any necessary DDM packages (including scripts, patterns, and so on) by selecting: **Settings** > **Import packages**.

13 Breakpoints

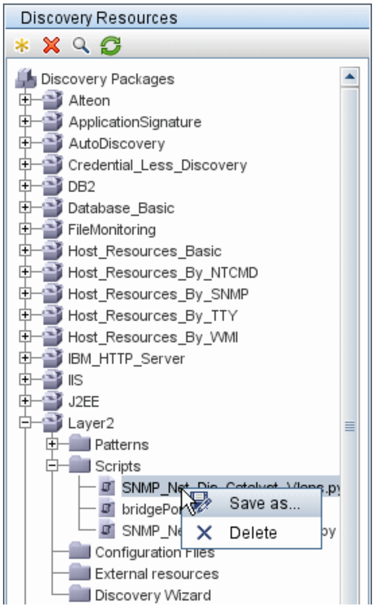
If you run Discovery Analyzer from the Python script, you can add breakpoints to your script.

Discovery and Dependency Mapping Code

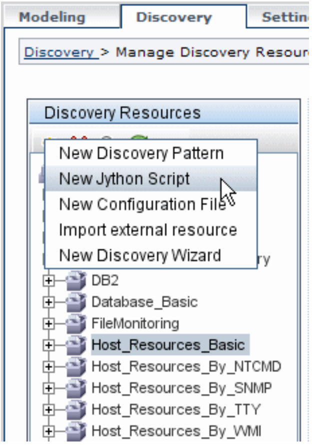
The actual implementation of connecting to the remote system, querying its data, and mapping it as CMDB data is performed by the DDM code. For example, the code contains the logic for connecting to a database and extracting data from it. In this case, the code expects to receive a JDBC URL, a user name, a password, a port, and so on. These parameters are specific for each instance of the database that answers the TQL query. You define these variables in the pattern (in the Trigger CI data) and when the job runs, these specific details are passed to the code for execution.

The pattern can refer to this code by a Java class name or a Jython script name. In this section we discuss writing DDM code as Jython scripts.

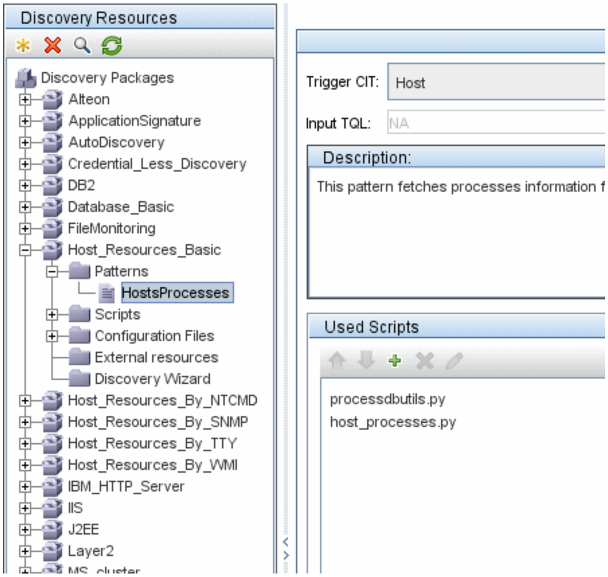
A pattern can contain a list of scripts to be used when running DDM. When creating a new pattern, you usually create a new script and assign it to the pattern. A new script includes basic templates, but you can use one of the other scripts as a template by right-clicking it and selecting **Save as**:



For details on writing new Jython scripts, see “Step 3: Create Jython Code” on page 328. You add scripts through the Manage Discovery Resources window:



The list of scripts are run one after the other, in the order in which they are defined in the pattern:



Trigger CIT: Host

Input TQL: NA

Description:

This pattern fetches processes information f

processdbutils.py
host_processes.py

Note: A script must be specified even though it is being used solely as a library by another script. In this case, the library script must be defined before the script using it. In this example, the processdbutils.py script is a library used by the last host_processes.py script. Libraries are distinguished from regular runable scripts by the lack of the DiscoveryMain() function.

Jython Libraries and Utilities

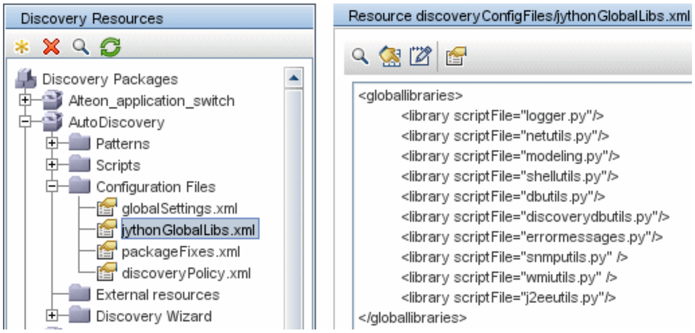
Several utility scripts are used widely in DDM patterns. These scripts are part of the AutoDiscovery package and are located under: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryScripts** with the other scripts that are downloaded to the Probe.

Note: The discoveryScript folder is created dynamically when the Probe begins working.

To use one of the utility scripts, add the following import line to the import section of the script:

```
import <script name>
```

The AutoDiscovery Python library contains Jython utility scripts. These library scripts are considered DDM's external library. They are defined in the jythonGlobalLibs.xml file (located in the **Configuration Files** folder).



```
<globallibraries>
        <library scriptFile="logger.py"/>
        <library scriptFile="netutils.py"/>
        <library scriptFile="modeling.py"/>
        <library scriptFile="shellutils.py"/>
        <library scriptFile="dbutils.py"/>
        <library scriptFile="discoverydbutils.py"/>
        <library scriptFile="errormessages.py"/>
        <library scriptFile="snmputils.py" />
        <library scriptFile="wmiutils.py" />
        <library scriptFile="j2eeutils.py"/>
</globallibraries>
```

Each script that appears in the jythonGlobalLibs.xml file is loaded by default at Probe startup, so there is no need to use them explicitly in the pattern definition.

This section includes the following topics:

- "logger.py" on page 363
- "modeling.py" on page 365
- "netutils.py" on page 365
- "shellutils.py" on page 365

logger.py

The **logger.py** script contains log utilities and helper functions for error reporting. You can call its debug, info, and error APIs to write to the log files. Log messages are recorded in **C:\hp\DDM\DiscoveryProbe\root\logs\probeMgr-patternsDebug.log**.

Messages are entered in the log file according to the debug level defined for the PATTERNS_DEBUG appender in the **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\probeMgrLog4j.properties** file. (By default, the level is DEBUG.) For details, see “Severity Levels” on page 68.

```
############################################################################
#################        PATTERNS_DEBUG log              ######################
############################################################################
log4j.category.PATTERNS_DEBUG=DEBUG, PATTERNS_DEBUG
log4j.appender.PATTERNS_DEBUG=org.apache.log4j.RollingFileAppender
log4j.appender.PATTERNS_DEBUG.File=C:/hp/DDM/DiscoveryProbe/root/logs/probe
Mgr-patternsDebug.log
log4j.appender.PATTERNS_DEBUG.Append=true
log4j.appender.PATTERNS_DEBUG.MaxFileSize=15MB
log4j.appender.PATTERNS_DEBUG.Threshold=DEBUG
log4j.appender.PATTERNS_DEBUG.MaxBackupIndex=10
log4j.appender.PATTERNS_DEBUG.layout=org.apache.log4j.PatternLayout
log4j.appender.PATTERNS_DEBUG.layout.ConversionPattern=<%d> [%-5p] [%t] -
%m%n
log4j.appender.PATTERNS_DEBUG.encoding=UTF-8
```

The info and error messages also appear in the Command Prompt console.

There are two sets of APIs:

- logger.<debug/info/warn/error>
- logger.<debugException/infoException/warnException/errorException>

The first set issues the concatenation of all its string arguments at the appropriate log level and the second set issues the concatenation as well as issuing the stack trace of the most recently-thrown exception, to provide more information, for example:

```
logger.debug('found the result')
logger.errorException('Error in discovery')
```

modeling.py

The **modeling.py** script contains APIs for creating hosts, IPs, process CIs, and so on. These APIs enable the creation of common objects and make the code more readable. For example:

```
ipOSH= modeling.createIpOSH(ip)
host = modeling.createHostOSH(ip_address)
member1 = modeling.createLinkOSH('member', ipOSH, networkOSH)
```

netutils.py

The **netutils.py** library is used to retrieve network and TCP information, such as retrieving operating system names, checking if a MAC address is valid, checking if an IP address is valid, and so on. For example:

```
dnsName = netutils.getHostName(ip, ip)
isValidIp = netutils.isValidIp(ip_address)
address = netutils.getHostAddress(hostName)
```

shellutils.py

The **shellutils.py** library provides an API for executing shell commands and retrieving the end status of an executed command, and enables running multiple commands based on that end status. The library is initialized with a Shell Client, and uses the client to run commands and retrieve results. For example:

```
ttyClient = clientFactory.createClient(Props)
clientShUtils = shellutils.ShellUtils(ttyClient)
if (clientShUtils.isWinOs()):
      logger.debug ('discovering Windows..')
```

Job and Pattern XML Formats

Jobs and patterns are saved in the CMDB in an XML format. The job name appears in the job's XML file and is referred to by the **id** attribute. Each job has a corresponding pattern which is referred to by the **patternId** attribute.

Example of Job XML

A job name is **CPUs by TTY**. Its corresponding pattern is **TTY_HR_CPU**:

```
<job id="CPUs by TTY" displayName="CPUs by TTY">
        <patternId>TTY_HR_CPU</patternId>
        <triggers>
               <trigger>shell_on_unix</trigger>
        </triggers>
        <schedule>
               <offsetElem>0</offsetElem>
               <generatedFromElem>1</generatedFromElem>
               <schedularType>scheduler_simple_Schedule</schedularType>
               <timeExpElem>Days_1</timeExpElem>
        </schedule>
</job>
```

Example of Pattern XML

The pattern name is **TTY_HR_CPU**. The pattern input is defined by the <**inputClass**> tag. The pattern output is defined by the <**discoveredClasses**> tag.

```
<pattern id="TTY_HR_CPU" description="Discover CPU on Unix boxes."
schemaVersion="7.0">
 <discoveredClasses>
      <discoveredClass>container_f</discoveredClass>
      <discoveredClass>cpu</discoveredClass>
 </discoveredClasses>
 <parameters />
 <taskInfo className="appilog.collectors.services.dynamic.core.DynamicService">
      <destinationInfo className="appilog.collectors.tasks.BaseDestinationData">
           <destinationData
name="ip_address">${SOURCE.application_ip}</destinationData>
           <destinationData
name="Protocol">${SOURCE.root_class}</destinationData>
           <destinationData
name="credentialsId">${SOURCE.credentials_id}</destinationData>
           <destinationData
name="language">${SOURCE.language:NA}</destinationData>
           <destinationData
name="codepage">${SOURCE.codepage:NA}</destinationData>
      </destinationInfo>
      <params
className="appilog.collectors.services.dynamic.core.DynamicServiceParams">
           <script>TTY_HR_CPU_Lib.py</script>
           <script>TTY_HR_CPU.py</script>
      </params>
 </taskInfo>
 <inputClass>shell</inputClass>
</pattern>
```

12

Supporting Multi-Lingual Locales

This chapter includes:

Concepts

Tasks

Reference

Supporting Multi-Lingual Locales Overview

Note: This functionality is available as part of Content Pack 3.00 or later.

The multi-lingual locale feature enables DDM to work across different operating system (OS) languages, and to enable appropriate customizations at runtime.

Previously, before Content Pack 3.00, DDM used statically-specified encoding to treat output from all network targets. However, this approach does not suit a multi-lingual IT network: to discover hosts with different OS languages, Probe administrators had to re-run DDM jobs manually several times with different job parameters each time. This procedure produced a serious overhead on network load but, even more, it avoided several key features of DDM, such as immediate job invocation on a trigger CI or automatic data refreshing in UCMDB by the Schedule Manager.

The following locale languages are supported by default: Russian, German. The default locale is English.

Determining the Character Set for Encoding

The suitable character set for decoding command output is determined at runtime. The multi-lingual solution is based on the following facts and assumptions:

1. It is possible to determine the OS language in a locale-independent way, for example, by running the **chcp** command on Windows or the **locale** command on Linux.
2. Relation Language-Encoding is well known and can be defined statically. For example, the Russian language has two of the most popular encoding: Cp866 and Windows-1251.
3. One character set for each language is preferable, for example, the preferable character set for Russian language is Cp866. This means that most of the commands produce output in this encoding.

4. Encoding in which the next command output is provided is unpredictable, but it is one of the possible encoding for a given language. For example, when working with a Windows machine with a Russian locale, the system provides the **ver** command output in Cp866, but the **ipconfig** command is provided in Windows-1251.

5. A known command produces known key words in its output. For example, the **ipconfig** command contains the translated form of the **IP-Address** string. So the **ipconfig** command output contains **IP-Address** for the English OS, IP-Адрес for the Russian OS, **IP-Adresse** for the German OS, and so on.

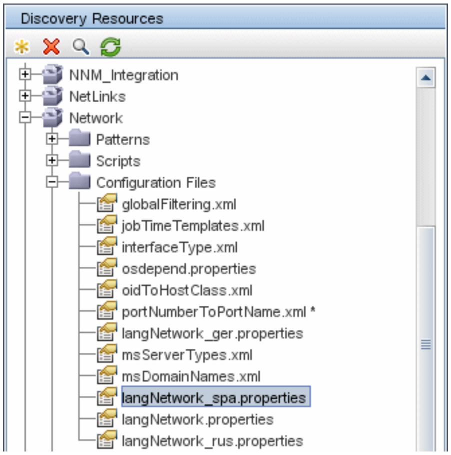
   Once it is discovered in which language the command output is produced (# 1), possible character sets are limited to one or two (# 2). Furthermore, it is known which key words are contained in this output (# 5).

   The solution, therefore, is to decode the command output with one of the possible encoding by searching for a key word in the result. If the key word is found, the current character set is considered the correct one.

Resource Bundles

A resource bundle is a file that takes a properties extension (***.properties**). A properties file can be considered a dictionary that stores data in the format of key = value. Each row in a properties file contains one key = value association. The main functionality of a resource bundle is to return a value by its key.

Resource bundles are located on the Probe machine: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\discoveryConfigFiles**. They are downloaded from the UCMDB server as any other configuration file. They can be edited, added, or removed, in the Manage Discovery Resources window. For details, see "Configuration File Pane" on page 243.



When discovering a destination, DDM usually needs to parse text from command output or file content. This parsing is often based on a regular expression. Different languages require different regular expressions to be used for parsing. For code to be written once for all languages, all language-specific data must be extracted to resource bundles. There is a resource bundle for each language. (Although it is possible that a resource bundle contain data for different languages, in DDM one resource bundle always contains data for one language.)

The Jython script itself does not include hard coded, language-specific data (for example, language-specific regular expressions). The script determines the language of the remote system, loads the proper resource bundle, and obtains all language-specific data by a specific key.

In DDM, resource bundles take a specific name format: <base_name>_<language_identifier>.properties, for example, langNetwork_spa.properties. (The default resource bundle takes the following format: <base_name>.properties, for example, langNetwork.properties.)

The base_name format reflects the intended purpose of this bundle. For example, **langMsCluster** means the resource bundle contains language-specific resources used by the MS Cluster DDM jobs.

The language_identifier format is a 3-letter acronym used to identify the language. For example, rus stands for the Russian language and ger for the German language. This language identifier is included in the declaration of the Language object.

Add Support for New Language

This task describes how to add support for a new language.

This task includes the following steps:

- ➤ "Add a Resource Bundle (*.properties Files)" on page 374
- ➤ "Declare and Register the Language Object" on page 374

1 Add a Resource Bundle (*.properties Files)

Add a resource bundle according to the job that is to be run. The following table lists the DDM jobs and the resource bundle that is used by each job:

Job	Base Name of Resource Bundle
File Monitor by Shell	langFileMonitoring
Host Resources and Applications by Shell	langHost_Resources_By_TTY, langTCP
Hosts by Shell using NSLOOKUP in DNS Server	langNetwork
Host Connection by Shell	langNetwork
Collect Network Data by Shell or SNMP	langTCP
Host Resources and Applications by SNMP	langTCP
Microsoft Exchange Connection by NTCMD, Microsoft Exchange Topology by NTCMD	msExchange
MS Cluster by NTCMD	langMsCluster

For details on bundles, see “Resource Bundles” on page 372.

2 Declare and Register the Language Object

To define a new language, add the following two lines of code to the **shellutils.py** script, that currently contains the list of all supported languages. The script is included in the AutoDiscoveryContent package. To view the script, access the Manage Discovery Resources window. For details, see “Manage Discovery Resources Window” on page 256.

a Declare the language, as follows:

```
LANG_RUSSIAN = Language(LOCALE_RUSSIAN, 'rus', ('Cp866', 'Cp1251'),
(1049,), 866)
```

For details on class language, see “API Reference” on page 377. For details on the Class Locale object, see http://java.sun.com/j2se/1.5.0/docs/api/java/util/Locale.html. You can use an existing locale or define a new locale.

b Register the language by adding it to the following collection:

```
LANGUAGES = (LANG_ENGLISH, LANG_GERMAN, LANG_SPANISH,
LANG_RUSSIAN, LANG_JAPANESE)
```

Define a New Job to Operate With Localized Data

This task describes how to write a new job that can operate with localized data.

Jython scripts usually execute commands and parse their output. To receive this command output in a properly decoded manner, you use the API for the **ShellUtils** class. For details, see “The DDM Web Service API” on page 289.

This code usually takes the following form:

```
client = Framework.createClient(protocol, properties)
shellUtils = shellutils.ShellUtils(client)
languageBundle = shellutils.getLanguageBundle (’langNetwork’, shellUtils.osLanguage,
Framework)
strWindowsIPAddress = languageBundle.getString(’windows_ipconfig_str_ip_address’)
ipconfigOutput = shellUtils.executeCommandAndDecode(’ipconfig /all’,
strWindowsIPAddress)
#Do work with output here
```

1 Create a client:

```
client = Framework.createClient(protocol, properties)
```

2 Create an instance of the **ShellUtils** class and add the operating system language to it. If the language is not added, the default language is used (usually English):

```
shellUtils = shellutils.ShellUtils(client)
```

During object initialization, DDM automatically detects machine language and sets preferable encoding from the predefined Language object. Preferable encoding is the first instance appearing in the encoding list.

3. Retrieve the appropriate resource bundle from **shellclient** using the **getLanguageBundle** method:

   ```
   languageBundle = shellutils.getLanguageBundle ('langNetwork', shellUtils.osLanguage,
   Framework)
   ```

4. Retrieve a keyword from the resource bundle, suitable for a particular command:

   ```
   strWindowsIPAddress = languageBundle.getString('windows_ipconfig_str_ip_address')
   ```

5. Invoke the **executeCommandAndDecode** method and pass the keyword to it on the **ShellUtils** object:

   ```
   ipconfigOutput = shellUtils.executeCommandAndDecode('ipconfig /all',
   strWindowsIPAddress)
   ```

   The ShellUtils object is also needed to link a user to the API reference (where this method is described in detail).

6. Parse the output as usual.

Decoding Commands Without a Keyword

The current approach for localization uses a keyword to decode all of the command output. For details, see step 4 on page 376 in “Define a New Job to Operate With Localized Data” on page 375.

However, another approach uses a keyword to decode the first command output only, and then decodes further commands with the character set used to decode the first command. To do this, you use the **getCharsetName** and **useCharset** methods of the **ShellUtils** object.

The regular use case works as follows:

1. Invoke the **executeCommandAndDecode** method once.
2. Obtain the most recently used character set name through the **getCharsetName** method.

3. Make **shellUtils** use this character set by default, by invoking the **useCharset** method on the **ShellUtils** object.
4. Invoke the **execCmd** method of **ShellUtils** one or more times. The output is returned with the character set specified in step 3 on page 377. No additional decoding operations occur.

Change the Default Language

If the OS language cannot be determined, the default one is used. The default language is specified in the **shellutils.py** file.

```
#default language for fallback
DEFAULT_LANGUAGE = LANG_ENGLISH
```

To change the default language, you initialize the DEFAULT_LANGUAGE variable with a different language. For details, see “Add Support for New Language” on page 373.

API Reference

This section includes:

- “The Language Class” on page 377
- “The executeCommandAndDecode Method” on page 378
- “The getCharsetName Method” on page 379
- “The useCharset Method” on page 379
- “The getLanguageBundle Method” on page 380
- “The osLanguage Field” on page 380

The Language Class

This class encapsulates information about the language, such as resource bundle postfix, possible encoding, and so on.

Fields

Name	Description
locale	Java object which represents locale.
bundlePostfix	Resource bundle postfix. This postfix is used in resource bundle file names to identify the language. For example, the **langNetwork_ger.properties** bundle includes a **ger** bundle postfix.
charsets	Character sets used to encode this language. Each language can have several character sets. For example, the Russian language is commonly encoded with the Cp866 and Windows-1251 encoding.
wmiCodes	The list of WMI codes used by the Microsoft Windows OS to identify the language. All possible codes are listed at http://msdn.microsoft.com/en-us/library/aa394239(VS.85).aspx (the OSLanguage section). One of the methods for identifying the OS language is to query the WMI class OS for the OSLanguage property.
codepage	Code page used with a specific language. For example, 866 is used for Russian machines and 437 for English machines. One of the methods for identifying the OS language is to retrieve its default codepage (for example, by the chcp command).

The executeCommandAndDecode Method

This method is intended to be used by business logic Jython scripts. It encapsulates the decoding operation and returns a decoded command output.

Arguments

Name	Description
cmd	The actual command to be executed.
keyword	The keyword to be used for the decoding operation.
framework	The Framework object passed to every executable Jython script in DDM.
timeout	The command timeout.
waitForTimeout	Specifies if client should wait when timeout is exceeded.
useSudo	Specifies if sudo should be used (relevant only for UNIX machine clients).
language	Enables specifying the language directly instead of automatically detecting a language.

The getCharsetName Method

This method return the name of the most recently used character set.

The useCharset Method

This method sets the character set on the ShellUtils instance, which uses this character set for initial data decoding.

Arguments

Name	Description
charsetName	The name of the character set, for example, windows-1251 or UTF-8.

See also “The getCharsetName Method” on page 379.

The getLanguageBundle Method

This method should be used to obtain the correct resource bundle. This replaces the following API:

```
Framework.getEnvironmentInformation().getBundle(…)
```

Arguments

Name	Description
baseName	The name of the bundle without the language suffix, for example, langNetwork.
language	The language object. The ShellUtils.osLanguage should be passed here.
framework	The Framework, common object which is passed to every executable Jython script in DDM.

The osLanguage Field

This field contains an object that represents the language.

Part V

Discovery and Dependency Mapping Security

13

Hardening DDM

This chapter provides information on hardening Discovery and Dependency Mapping (DDM).

This chapter includes:

Concepts

Tasks

Hardening Discovery and Dependency Mapping Overview

Discovery credentials entered in the Set Up Discovery Probes window are saved in an encrypted file termed a domain scope document (DSD). This DSD contains discovery domain data. Each discovery domain entry in the document contains the network scope for the domain's Probes and the credentials the Probes may use when communicating with remote machines.

Note: Security features related to HP Business Availability Center user management—for example, authentication and authorization—are not discussed here.

This section includes the following topics:

- "Basic Security Assumptions" on page 384
- "Credentials Encryption Management" on page 384
- "HTTPS\SSL Configuration" on page 385

Basic Security Assumptions

Note the following security assumptions:

- You have secured the HP Business Availability Center Gateway Server and Probe file systems for authorized access only.
- You have secured the Gateway Server JMX console to enable access to BAC system administrators only, preferably through localhost access only.

Credentials Encryption Management

Note the following guidelines for managing credential encryption:

- The **domainScopeDocument** (DSD) file is encrypted using standard, symmetric encryption. The DSD file is encrypted during transfer and at its location (the key is stored on the server database and the Probe file system). The encryption uses an AES algorithm with a default key of 192 bytes (but you can decide the length of the encryption/decryption key and other security parameters). For details on changing the key size, see "Update an Encryption Key" on page 391.
- A default, symmetric key is distributed with the BAC installation. As all default keys are identical, you should replace this key with a locally-generated key.
- The DSD file is exportable and importable in encrypted file form. To import a file you should supply the matching key used for the encryption of the file. Perform the import and export operations through the server's JMX console (the Discovery Manager service). For details, see "Export and Import the domainScopeDocument (DSD) File in Encrypted Format" on page 394.
- You can exchange the key while the system is up to keep the consistency of the system without losing any data. This is managed through the server's JMX console. For details, see "Generate or Update the Encryption Key" on page 389.
- When a key is updated, you can automatically distribute the new key to the Probes. This option is easier to deploy but is considered less secure. The new key is encrypted using the old key. To achieve better security, you can change the key manually. For details, see "Generate a New Encryption Key" on page 390.
- The key for DSD encryption is itself encrypted using DPAPI and is stored on the Probe and BAC server file systems (in encrypted format–and not in clear text). DPAPI relies on the Windows user password in the encryption process. Therefore, to ensure that the Probe can read the key, the Probe should always run under the same Windows user (it is possible to change the user password). (DPAPI is a standard method to protect confidential data—such as certificates and private keys—on Windows.)

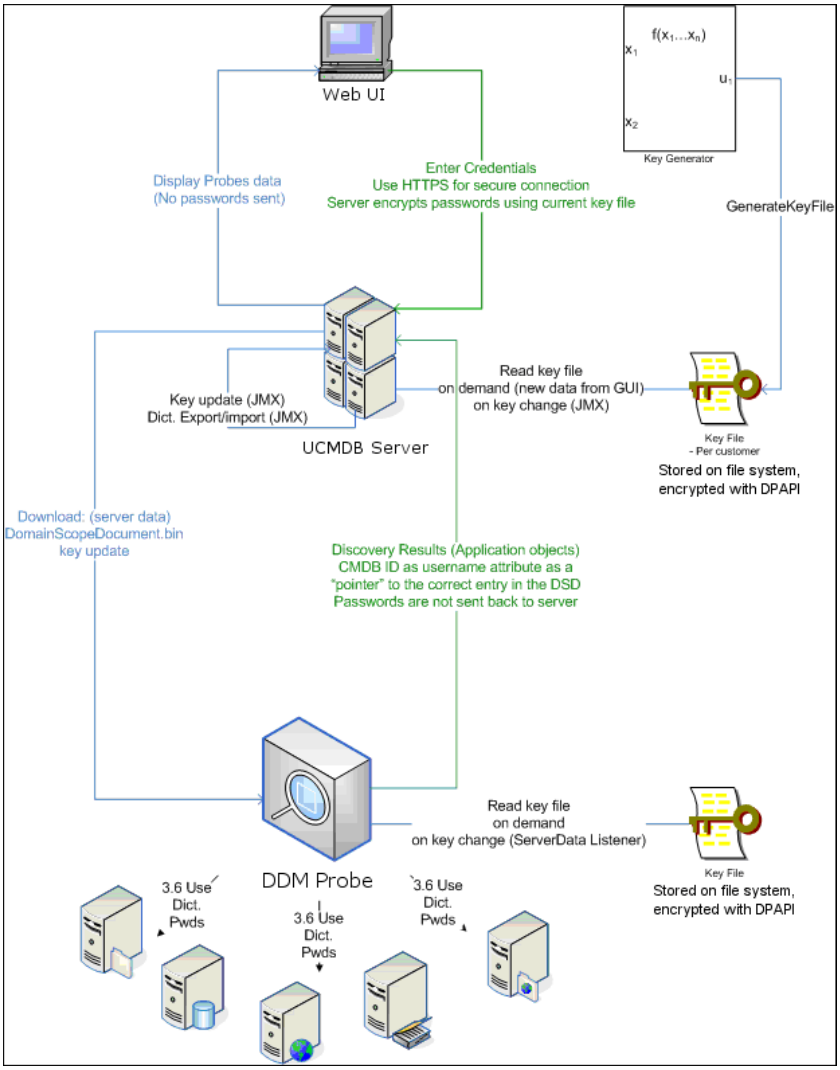
HTTPS\SSL Configuration

You can configure communication between the HP Business Availability Center Gateway Server and the DDM Probe to use HTTPS\SSL. This enables better DSD security during transit.

For details on the procedure to configure HP Business Availability Center to work with SSL, see “Using SSL in HP Business Availability Center” in the *HP Business Availability Center Hardening Guide* PDF.

Note: As a result of using SSL, other aspects (for example, discovery tasks and gathered results) of the HP Business Availability Center product may become more secure.

The following illustration depicts a hardened system.



Manage the Storage of Credentials

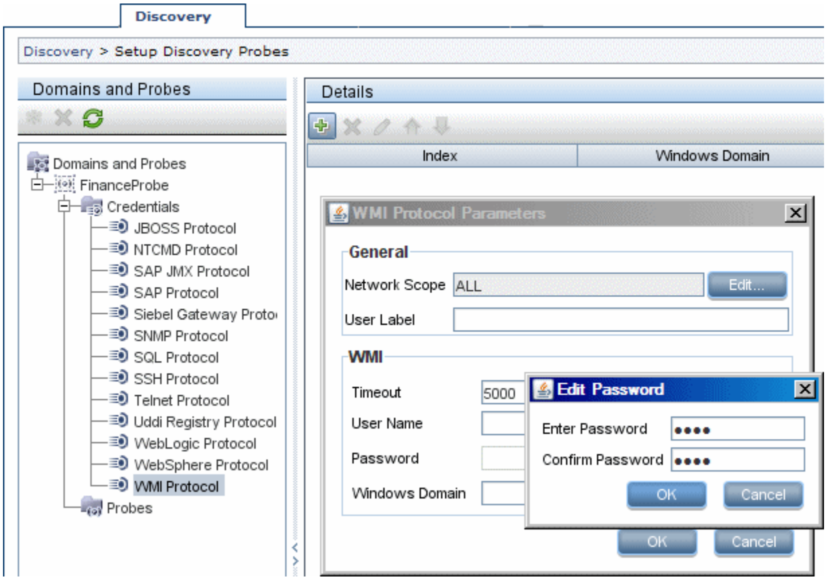
This section explains how to manage the DSD file.

This section includes the following tasks:

- "View Credentials Information (Data Direction: Server to HP Business Availability Center)" on page 388
- "Update Credentials (Data Direction: HP Business Availability Center to Server)" on page 389

1 View Credentials Information (Data Direction: Server to HP Business Availability Center)

Passwords are not sent from the Server to the application. That is, HP Business Availability Center displays asterisks (*) in the password field, regardless of content:



Discovery > Setup Discovery Probes

Network Scope: ALL

Timeout: 5000

Enter Password: ●●●●

Confirm Password: ●●●●

2 Update Credentials (Data Direction: HP Business Availability Center to Server)

- The communication in this direction is not encrypted, therefore you should connect to the BAC Gateway Server using https\SSL, or ensure connection through a trusted network.

  Although the communication is not encrypted, passwords are not being sent as clear text on the network. They are encrypted using a default key and, therefore, it is highly recommended to use SSL for effective confidentiality in transit.

- The password field is limited to 40 characters. The length of the password is not limited in other ways since it is saved only in a file.
- You can use special characters and non-English characters as passwords.

Generate or Update the Encryption Key

You can generate or update an encryption key. In each case DDM creates a new encryption key based on parameters that you supply (key length, extra PBE cycles, JCE provider). You can also disable DPAPI encryption.

Note:

- The difference between the methods used to create a key (generateEncryptionKey) and to update a key (changeEncryptionKey) is that generateEncryptionKey creates a new, random encryption key, while changeEncryptionKey imports an encryption key whose name you provide.
- Only one encryption key can exist on a system, no matter how many Probes are installed.

This section includes the following tasks:

- "Generate a New Encryption Key" on page 390
- "Update an Encryption Key" on page 391
- "Retrieve Encryption Key File Name" on page 392
- "Disable DPAPI Encryption" on page 393
- "Define Several JCE Providers" on page 393

Generate a New Encryption Key

You can generate a new key to be used by the BAC Server and Probe for encryption/decryption. DDM replaces the old key with the new generated key, and distributes this key among the Probes.

To generate a new encryption key through the JMX Console:

1. Launch the Web browser and enter the server address, as follows: http://localhost.<domain_name>:8080/jmx-console.

   You may have to log in with a user name and password.

2. Under MAM, click **service=Discovery manager** to open the JMX MBEAN View page.

3. Locate the **generateEncryptionKey()** operation.

   - In the **customerId** parameter box, enter **1** (the default).
   - **keySizeInBits**. The length of the encryption key (the length can be 128, 192, or 256).
   - **usePBE**. **true**: use additional PBE hash cycles. **false**: do not use additional PBE hash cycles.
   - **jceVendor**. You can choose to use a non-default JCE provider. If the box is empty, the default provider is used.
   - **autoUpdateProbe**. Leave as **True** for the server to automatically distribute the new key to the Probes.

     Select **False** to distribute the new key manually using the Probe JMX console.

Important:

Make sure that the Probe is up and connected to the server. If the Probe goes down, the key cannot reach the Probe.

If you have changed the key before the Probe goes down, once the Probe is up again, the key is sent again to the Probe. However, if you have changed the key more than once before the Probe goes down, you must change the key manually through the JMX console. (Select **False**).

---

4 Click **Invoke** to generate the encryption key.

Update an Encryption Key

You use the changeEncryptionKey method to import an encryption key.

To update an encryption key through the JMX Console:

1 Launch the Web browser and enter the server address, as follows: http://localhost.<domain_name>:8080/jmx-console.

2 You may have to log in with a user name and password.

3 Under MAM, click **service=Discovery manager** to open the JMX MBEAN View page.

4 Locate the **changeEncryptionKey()** operation.

- ➤ In the **customerId** parameter box, enter **1** (the default).
- ➤ **newKeyFileName**. Enter the name of the new key.
- ➤ **keySizeInBits**. The length of the encryption key (the length can be 128, 192, or 256).
- ➤ **usePBE**. **true**: use additional PBE hash cycles. **false**: do not use additional PBE hash cycles.
- ➤ **jceVendor**. You can choose to use a non-default JCE provider. If the box is empty, the default provider is used.
- ➤ **autoUpdateProbe**. Leave as **True** for the server to automatically distribute the changed key to the Probes.

Select **False** to distribute the changed key manually using the Probe JMX console.

Important:

Make sure that the Probe is up and connected to the server. If the Probe goes down, the key cannot reach the Probe.

If you have changed the key before the Probe goes down, once the Probe is up again, the key is sent again to the Probe. However, if you have changed the key more than once before the Probe goes down, you must change the key manually through the JMX console. (Select **False**).

---

5 Click **Invoke** to generate and update the encryption key.

Retrieve Encryption Key File Name

When you change an encryption key on the server (by using the **changeEncryptionKey** method), you may not want the new key to be downloaded automatically to the Probes because of security concerns. You can download the encryption key file to a Probe manually.

To prevent automatic download, run the importEncryptionKey method:

1 Place the encryption key file in the **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\binaryData** directory.

2 Launch the Web browser and enter the Probe address, as follows: http://localhost.<domain_name>:1977/.

You may have to log in with a user name and password.

3 Under the Probe domain, click **type=MainProbe** to open the MBean View page.

4 Locate the **importEncryptionKey** method.

5 Enter the name of the encryption key file that resides in the **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\binaryData** directory.

This file contains the key to be imported.

6 Click the **importEncryptionKey** button.

Disable DPAPI Encryption

The key is encrypted with DPAPI on the file system. You can disable this encryption.

To disable encryption with DPAPI:

1 Launch the Web browser and enter the server address, as follows: http://localhost.<domain_name>:8080/jmx-console.

You may have to log in with a user name and password.

2 Under **MAM**, click **service=Discovery manager** to open the JMX MBEAN View page.

3 Locate the **setDPApiUsage()** operation and enter the following properties:

- In the **customerId** parameter box, enter **1** (the default).
- **useDPAPI**. **true**: use DPAPI. **false**: do not use DPAPI.

4 Click **Invoke** to disable the DPAPI encryption.

Define Several JCE Providers

When generating an encryption key through the JMX Console, you can define several JCE providers with the **changeEncryptionKey** and **generateEncryptionKey** methods. To change the default JCE provider:

1 Register the JCE provider jar files in the **$JRE_HOME/lib/ext** directory.

2 Copy the jar files to the following directory:

For the **UCMDB server**: $JRE_HOME resides in the **C:\hp\UCMDB\UCMDBServer\jre** directory.

For the **DDM Probe**: $JRE_HOME resides in the **C:\hp\DDM\DiscoveryProbe\jre** directory.

3 Add the provider class at the end of the provider list in the **$JRE_HOME\lib\security\java.security** file.

4 Update the **local_policy.jar** and **US_export_policy.jar** files to include unlimited JCE policies. You can download these jar files from the Sun site.

5 Restart the UCMDB server and the DDM Probe.

6 Locate the JCE vendor field for **changeEncryptionKey** or **generateEncryptionKey** method, and add the name of the JCE provider.

Export and Import the domainScopeDocument (DSD) File in Encrypted Format

You can export and import DSD files in encrypted format. (You would probably import a DSD file during recovery following a system crash or during upgrade.)

- **When exporting the DSD file**, you must enter a password (of your choosing). The file is encrypted with this password. The encryption key used for storing the DSD file in the UCMDB database is no longer used.
- **When importing the DSD file**, you must use the same password that was defined when the DSD file was exported.

Important: If you exported the domainScopeDocument file from UCMDB version 8.02, to import the file to your present version, copy the password from the contents of the key.bin file located on the version 8.02 system.

To export or import a DSD file:

a Launch the Web browser and enter the following address: http://localhost.<domain_name>:8080/jmx-console.

You may have to log in with a user name and password.

b Under MAM, click **service=Discovery manager** to open the JMX MBEAN View page.

c Locate the **ExportDomainScopeDocument** or **importDomainScopeDocument** operation and enter the existing file name and password.

d Click **Invoke** to export or import the domainScopeDocument file.

The location of the saved domainScopeDocument file is the **C:\hp\UCMDB\UCMDBServer\root\lib\server\discovery\<customer_dir>** directory.

14

Hardening the DDM Probe

This chapter provides information on hardening the Discovery and Dependency Mapping (DDM) Probe.

For the procedure for enabling mutual SSL authentication on the DDM Probe, see “Enable SSL Between BAC Running an Internal UCMDB and the DDM Probe with Mutual Authentication” on page 404.

This chapter includes:

Tasks

Harden the MySQL Database

You can run a script to set the user password for the DDM Probe's MySQL database.

Script name: **set_dbuser_password.cmd**

Script location: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors**

You can use this script:

- Manually, whenever you need to either set or change the password: run the file at the command prompt, or double-click the file.
- Automatically, when you run the **delCollectors** script to clean up the Probe's database: the **mamprobe** user is recreated with the password defined in the **set_dbuser_password.sql** script.

To set the user password:

1. Open the following file in a text editor: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\set_dbuser_password.sql**.
2. Change **PASSWORD('')** to **PASSWORD('YOUR_PASSWORD')** in both the places that it appears.
3. Run the **set_dbuser_password.cmd** script.
4. Enter the password in the following files:

   a. **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties**

      Add the password to the following properties: **appilog.agent.probe.jdbc.pwd** and **appilog.agent.local.jdbc.pwd**

   b. **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\probeManager\probeMgr-quartz.properties**

      Add the password to the following properties: **org.quartz.dataSource.QUARTZ_DB.password**

5. Open the DDM application and access the **SQL-MySql.properties** file (**Admin** > **Universal CMDB** > **Discovery** > **Manage Discovery Resources** > **NetLinks** > **Configuration Files**).
6. Locate the **JDBC.Password=** property and enter the password.

To allow access to the DDM Probe's MySQL database from the local machine only:

Run the following script in a command prompt or by double-clicking it: **remove_remote_user_access.cmd**. Any user trying to connect from a remote computer is denied access.

Enable SSL on the DDM Probe with Basic Authentication

Important: The following method of enabling SSL on the DDM Probe (basic authentication security) is deprecated. It is recommended to use mutual authentication security, as it is a much more effective method of security. For details, see "Enable SSL Between BAC Running an Internal UCMDB and the DDM Probe with Mutual Authentication" on page 404.

To set basic authentication:

1 Locate the following file: **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties**.

2 Remove the comment markers (#) from the following properties, and enter the relevant credentials:

```
appilog.agent.Probe.BasicAuth.Realm=
appilog.agent.Probe.BasicAuth.User=
appilog.agent.Probe.BasicAuth.Pwd=
```

The credentials should match those defined on the BAC server.

Connect the DDM Probe by Reverse Proxy

Perform the following procedure to connect the DDM Probe by reverse proxy.

Note: Enabling mutual authentication when using SSL between the UCMDB server and the DDM Probe is not supported when the connection is made by reverse proxy.

To configure the DDM Probe to work against a reverse proxy:

1 Edit the **discoveryProbe.properties** file (located in <**DDM Probe root directory**>**\hp\DDM\DiscoveryProbe\root\lib\collectors**).

2 Set the **serverIP** property to the reverse proxy server's IP or DNS name.

3 Save the file.

The following proxy server configuration is required if DDM Probes only are connected via a reverse proxy to HP Business Availability Center:

Requests for... on the Reverse Proxy Server	Proxy Request to be Served by:
/mam-collectors/*	http://[HP Universal CMDB server]/mam-collectors/*

The following configuration is required if a SOAP adapter is used for replication via a reverse proxy to a secure (hardened) HP Universal CMDB:

Requests for... on the Reverse Proxy Server	Proxy Request to be Served by:
/axis2/*	http://[HP Universal CMDB server]/axis2/*

Connecting the DDM Probe and Web Clients by Reverse Proxy

The following configuration is required if both DDM Probes and application users are connected via a reverse proxy to HP Business Availability Center:

Requests for... on the Reverse Proxy Server	Proxy Request to be Served by:
/mam/*	[HP Universal CMDB server]/mam/*
/mam_images/*	[HP Universal CMDB server]/mam_images/*
/mam-collectors/*	[HP Universal CMDB server]/mam-collectors/*
/ucmdb/*	[HP Universal CMDB server]/ucmdb/*
/site	[HP Universal CMDB server]/site/*

Enforce Login to the DDM Probe's JMX Console

You can prevent unauthenticated access to the DDM Probe's JMX console. To access the Probe's JMX console, users must log in with a user name and password. The following procedure explains how to implement the user name and password.

1 Stop the DDM Probe service.

2 Open the **DiscoveryProbe.properties** file (located in the **root\lib\collector\probemanager** folder) in a text editor, and update the following values from:

```
# Authentication data for the probe JMX
# Values can be empty to represent non-values.
appilog.agent.Probe.JMX.BasicAuth.User=
appilog.agent.Probe.JMX.BasicAuth.Pwd=
```

to:

```
# Authentication data for the probe JMX
# Values can be empty to represent non-values.
appilog.agent.Probe.JMX.BasicAuth.User=admin
appilog.agent.Probe.JMX.BasicAuth.Pwd=admin
```

3 Test the result: In the Web browser, enter **http://<DDM Probe machine name or IP address>:1977**. A user name and password dialog box should be displayed.

Control the Location of the domainScopeDocument File

The Probe's file system holds (by default) both the encryption key and the domainScopeDocument file. Each time the Probe is started, the Probe retrieves the domainScopeDocument file from the server and stores it on its file system. To prevent unauthorized users from obtaining these credentials, you can configure the Probe so that the domainScopeDocument file is held in the Probe's memory and is not stored on the Probe file system.

To control the location of the domainScopeDocument file:

a Open **C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties** and change:

```
appilog.collectors.storeDomainScopeDocument=true
```

to

```
appilog.collectors.storeDomainScopeDocument=false
```

The Probe Gateway and Probe Manager serverData folders no longer contain the domainScopeDocument file.

For details on using the domainScopeDocument file to harden DDM, see "Manage the Storage of Credentials" on page 388.

b Restart the Probe.

15

Using SSL with the DDM Probe

This chapter includes:

Concepts

Tasks

Using SSL with the DDP Probe Overview

This chapter describes how to configure your HP Business Availability Center platform with the Discovery and Dependency Mapping Probe and Staging Data Replicator to support communication using the Secure Sockets Layer (SSL) channel.

For introductory and general information on configuring HP Business Availability Center and its data collectors to support SSL, see "Using SSL in HP Business Availability Center" in the *HP Business Availability Center Hardening Guide* PDF.

Enable SSL Between BAC Running an Internal UCMDB and the DDM Probe with Mutual Authentication

You can set up authentication for both the DDM Probe and BAC running an internal UCMDB, with certificates. The certificate for each side is sent and authenticated before the connection is established.

Important: The following method of enabling SSL on the DDM Probe replaces the procedure for basic authentication, which is deprecated. For details on basic authentication, see "Enable SSL on the DDM Probe with Basic Authentication" on page 399.

This section includes the following topics:

- "Prerequisites" on page 404
- "Enable Mutual Certificate Authentication" on page 404

Prerequisites

Set up the BAC server with an internal UCMDB, running in SSL. Client certificates are required.

Enable Mutual Certificate Authentication

If the certificate used by the HP Business Availability Center Web server is issued by a well-known Certificate Authority (CA), it is most likely that you do not have to perform the following procedure. To validate trust, try connecting to the Web server using SSL and check whether the certificate is already trusted.

During authentication, BAC running an internal UCMDB sends its certificate to the DDM Probe client machine, and the DDM Probe sends its certificate to BAC running an internal UCMDB.

1. Download the required certificate, encoded in base-64, and save it as **c:\cacert.cer**:

   - If BAC is using a CA-signed certificate: download the CA certificate.
   - If BAC is using a self-signed certificate: download the BAC self-signed certificate.

2. Import the Certificate Authority certificate into the DDM Java truststore by running the following command:

   ```
   C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -import -trustcacerts -alias ddmTrustedCA -
   keystore ..\lib\security\cacerts -file c:\cacert.cer
   ```

   - Type the following keystore password: **changeit**
   - When asked **Trust this certificate?**, enter **yes**.

3. Create a keystore by running the following command:

   ```
   C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -genkey -keyalg RSA -alias ddmkey -
   keystore
   C:\hp\DDM\DiscoveryProbe\root\lib\security\client.keystore
   ```

   - Choose your password and enter your details.

     Important: Enter the full hostname for **first and last name**.

   - When asked, **Is CN=... correct?** type **yes**.
   - Press ENTER to set the same password for the key.

4. Create a certificate request for CA to sign by running the following command:

   ```
   C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -certreq -alias ddmkey -file c:\ddm.csr  -
   keystore
   C:\hp\DDM\DiscoveryProbe\root\lib\security\client.keystore
   ```

   - Enter the keystore password.

5. Submit the **c:\ddm.csr** file to your Certificate Authority and acquire a signed client certificate in base-64 encoding.

6 Import the CA certificate into the keystore by running the following command:

```
C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -import -alias ddmTrustedCA -file
c:\cacert.cer -keystore
C:\hp\DDM\DiscoveryProbe\root\lib\security\client.keystore
```

➤ Enter the keystore password and when asked **Trust this certificate?**, enter **yes**.

7 Import the client certificate into the keystore by running the following command:

```
C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -import -alias ddmkey -file
c:\<SIGNED_CERT> -keystore
C:\hp\DDM\DiscoveryProbe\root\lib\security\client.keystore
```

Note: <**SIGNED_CERT**> is the full path to the certificate acquired in step 5 on page 405 above.

➤ Make sure that the output message is **Certificate** reply **was installed in keystore**.

8 List the contents of the keystore by running the following command:

```
C:\hp\DDM\DiscoveryProbe\jre\bin>keytool -list -keystore
C:\hp\DDM\DiscoveryProbe\root\lib\security\client.keystore
```

➤ Enter the keystore password.

➤ Verify that the output includes both **keyEntry** and **trustedCertEntry**.

9. Change the **ssl.properties** file, located in the **C:\HP\DDM\DiscoveryProbe\root\lib\security** folder:

   - Update the keystore and truststore file names to point to the files you created previously:

     ```
     # Path to Keystore and Truststore files
     javax.net.ssl.keyStore=C:\\hp\\DDM\\DiscoveryProbe\\root\\lib\\security\\client.keyst
     ore
     javax.net.ssl.trustStore=C:\\hp\\DDM\\DiscoveryProbe\\jre\\lib\\security\\cacerts
     ```

     (Note the double backslashes.)

   - Update the keystore and truststore passwords:

     - You encrypt the password through the Probe's JMX console: Launch a Web browser and enter the following address: **http://<DDM Probe machine name or IP address>:1977**. If you are running the DDM Probe locally, enter **http://localhost:1977**.

       You may have to log in with a user name and password.

     - Locate the **Type=MainProbe** service and click the link to open the JMX MBEAN View page.
     - Locate the **getEncryptedKeyPassword** operation.
     - Enter your keystore or truststore password in the **Key Password** field and click **getEncryptedKeyPassword**.
     - Open the **ssl.properties** file in the following folder: **C:\hp\DDM\DiscoveryProbe\root\lib\security**\.
     - Copy and paste the encrypted password (numbers separated by commas, for example, 1,2,3,4,5) into the relevant keystore or truststore line of the **ssl.properties** file.
     - Save the file.

10. Update the C:\hp\DDM\DiscoveryProbe\root\lib\collectors\DiscoveryProbe.properties file:

    - Change the **appilog.agent.probe.protocol** parameter to **HTTPS**.
    - Make sure the **serverPortHttps** value is **443**.

11. Restart the DDM Probe.

Configure SSL from the Discovery Probe to the Gateway Server

When a session is started between the Discovery Probe and the Gateway Server, the Gateway Server sends the probe a server-side certificate that was issued by a Certification Authority (CA) recognized by the Gateway Server. The Discovery Probe engine should be configured to trust the certificate or the CA that issued it, and to communicate via SSL.

To configure the Discovery Probe to connect to the Gateway Server using SSL:

1 Prerequisite: Configure HP Business Availability Center to use SSL.

2 Prerequisite: Install the Discovery Probe. During installation, enter the name of the HP Business Availability Center Gateway server to which the Probe must report results.

3 If you are working with the Certificate Authority, download the current Certificate Authority certificate to your DDM Probe server. Save it to a file, for example, C:\ca.cer.

4 Import this certificate into the DDM Probe JVM: **C:\hp\DDM\DiscoveryProbe\jre\bin** with the following values:

```
keytool -import -trustcacerts -alias <your alias> -keystore ..\lib\security\cacerts -file <file path and name>
```

5 Enter the password and click **Yes** to confirm.

6 Set the connection parameters in the DDM Probe.

➤ Open the file **%discovery root%\root\lib\collectors\DiscoveryProbe.properties**.

➤ Configure the URL of the HP Business Availability Center server:

```
serverName = <HP Business Availability Center Gateway server domain name>
```

Note: The SSL connection may fail if an IP address is used instead of domain name.

- Configure the port number to use for HTTPS:

  ```
  # Ports used for HTTP/s traffic
  #serverPort = 80
  serverPortHttps = 443
  ```

- Set the schema to be used by the Agent to HTTPS:

  ```
  # Can be either HTTP or HTTPS
  appilog.agent.probe.protocol = HTTPS
  ```

- Set the name of the HP Business Availability Center server:

  ```
  # Name of the Server machine to which this probe reports
  serverName = <server name either of the reverse proxy or the Gateway server>
  ```

7 Restart the Discovery Probe.

Index