HP Universal CMDB

for the Windows and UNIX operating systems

Software Version: 8.01 or later

Discovery and Dependency Mapping Content Pack 3.00

Document Release Date: July 2009
Software Release Date: July 2009

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

Trademark Notices

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

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

Java™ is a US trademark of Sun Microsystems, Inc.

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

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

Unix® is a registered trademark of The Open Group.

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

Welcome to This Guide

This guide describes the contents of DDM Content Pack 3.00.

This chapter includes:

How This Guide Is Organized

The guide contains the following chapter:

Chapter 1 DDM Content Pack 3.00 Introduction

This section provides general information about DDM Content Pack 3.00.

Chapter 2 DDM Content Pack 3.00 Installation for Windows

This section explains how to install the files needed to update HP Universal CMDB version 8.01 or later with DDM Content Pack 3.00, on a Windows system.

Chapter 3 Supporting Multi-Lingual Locales

This section explains how to set up DDM to discover system components in a multi-lingual locale environment.

Chapter 4 SQL Server Discovery by OS Credentials

This section explains how DDM uses operating system credentials to discover SQL Server elements.

Chapter 5 Class Model Enhancements

This section describes changes to the class model.

Who Should Read This Guide

This guide is intended for the following users of HP Universal CMDB:

- HP Universal CMDB administrators
- HP Universal CMDB platform administrators
- HP Universal CMDB application administrators
- HP Universal CMDB data collector administrators

Readers of this guide should be knowledgeable about enterprise system administration, have familiarity with ITIL concepts, and be knowledgeable about HP Universal CMDB.

Getting More Information

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

Documentation Updates

HP Software is continually updating its product documentation with new information.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to the HP Software Product Manuals Web site (http://h20230.www2.hp.com/selfsolve/manuals).

This *Discovery and Dependency Mapping Content Pack 3.00* guide is part of the HP Universal CMDB documentation for version 8.01 or later. For details on the DDM application, see the *Discovery and Dependency Mapping Guide* PDF.

1

DDM Content Pack 3.00 Introduction

This chapter introduces Discovery and Dependency Mapping (DDM) Content Pack 3.00.

This chapter includes:

Concepts

Overview

Note the following:

- Each DDM Content Pack includes all content from previous content packs.
- Once a content pack has been installed, existing packages are overridden. Therefore, you must back up any changed packages under a new name before installing a new content pack. For details, see "Back Up All DDM Packages" on page 20.
- If you made changes to any patterns or scripts of default jobs, you must back them up under a new name before installing a new content pack.

2

DDM Content Pack 3.00 Installation for Windows

This chapter describes the procedures to be performed to install Discovery and Dependency Mapping (DDM) Content Pack 3.00 on a Windows system.

This chapter includes:

Tasks

Install DDM Content Pack 3.00

The following procedure explains how to install DDM Content Pack 3.00.

This task includes the following sections:

- "Prerequisites" on page 14
- "Installation" on page 14

Prerequisites

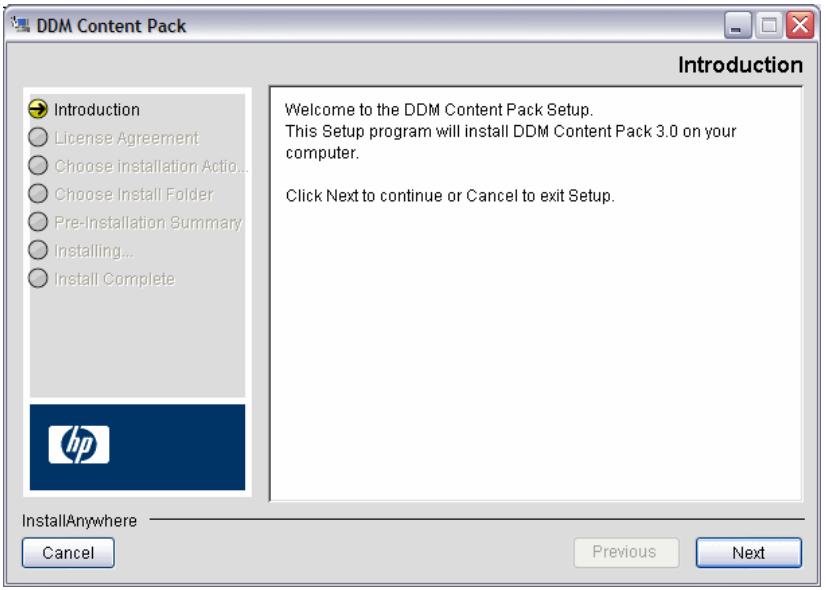
- **Important**: Before deploying DDM Content Pack 3.00 to the UCMDB version 8.0x server, you must restart the server, for the following reason: When UCMDB version 8.0x is installed as a patch on UCMDB version 8.00, packages in the patch_packages folder are deployed on the server at the next restart of the server, and override existing packages. If you deploy DDM Content Pack 3.00 packages before restarting the server, they are also overridden when you restart the version 8.0x server.
- The UCMDB server must be running when you install the Content Pack.
- During installation, Setup may restart the Probe, to load the new content jar file.
- When working with HP Business Availability Center in a distributed deployment, install DDM Content Pack 3.00 on the Data Processing Server.

Installation

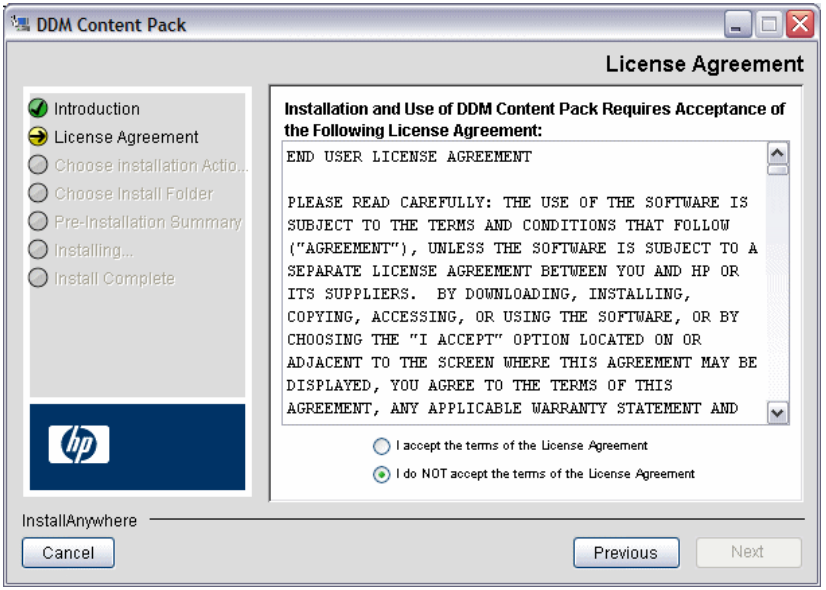
Perform the following procedure to install DDM Content Pack 3.00 on a Windows operating system:

1. Download **DDMContent_v30_win32.exe**. Download the file from the following FTP site: ftp://ddm_cp2:Brevard4@15.192.32.69/DDM_Content_Pack_300/
2. Run the **DDMContent_v30_win32.exe** file on the UCMDB Server machine.

The Introduction dialog box opens:

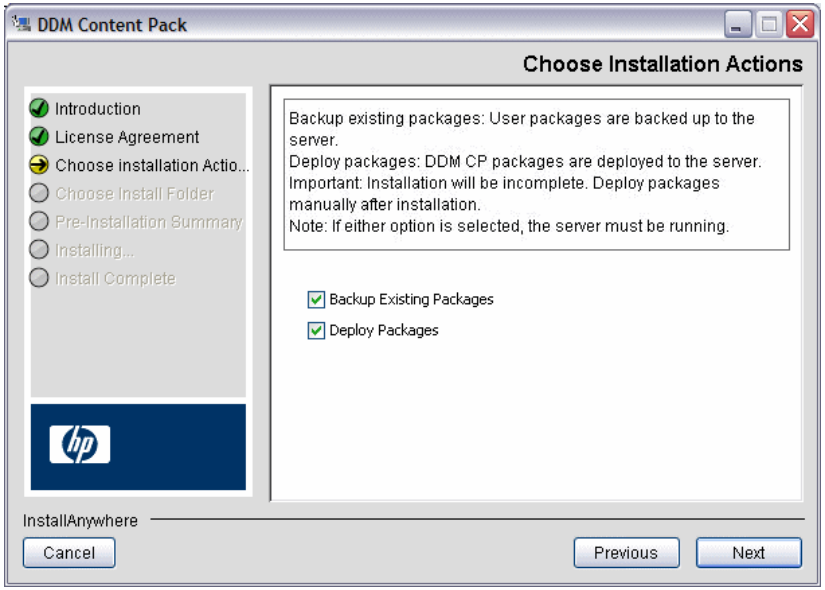


3 Click **Next** to open the License Agreement dialog box.



4 Accept the terms of the license and click **Next**.

5 The Choose Installation Actions dialog box opens:



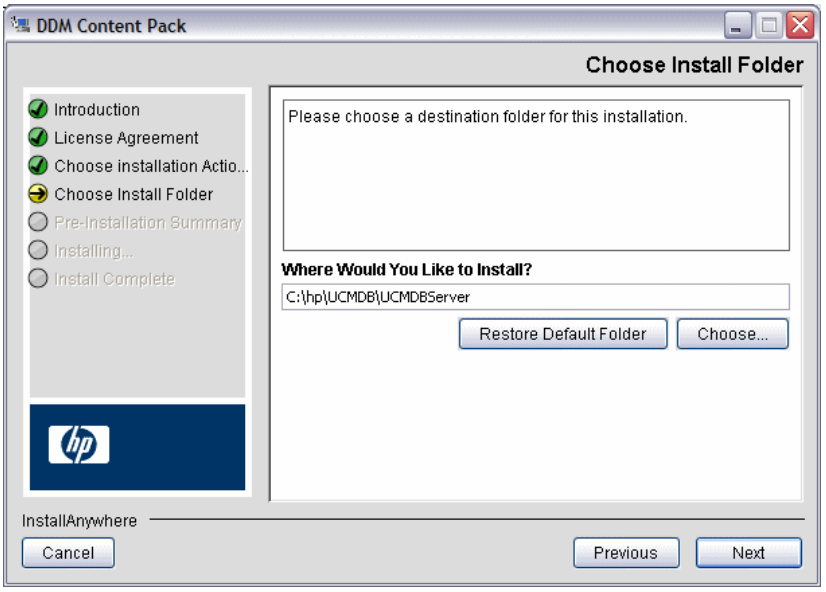
Choose the action the Installer should perform:

Note: If you select either or both of these options, verify that the server is running during installation.

- **Backup Existing Packages**: All user-defined packages are automatically backed up to a dedicated folder on the server: C:\hp\UCMDB\UCMDBServer\root\lib\ddm_content_packages_backup_before_3.0
- **Deploy Packages**: The DDM Content Pack packages are automatically deployed to the server.

Important: If you clear this option, installation is incomplete. You will need to run a manual procedure later to deploy packages to the server. For details, see "Deploy Packages" on page 20.

If Setup cannot verify the location of the HP Universal CMDB server directory or if you cleared the **Deploy packages** check box in the Choose Installation Actions dialog box, the Choose Install Folder dialog box opens:

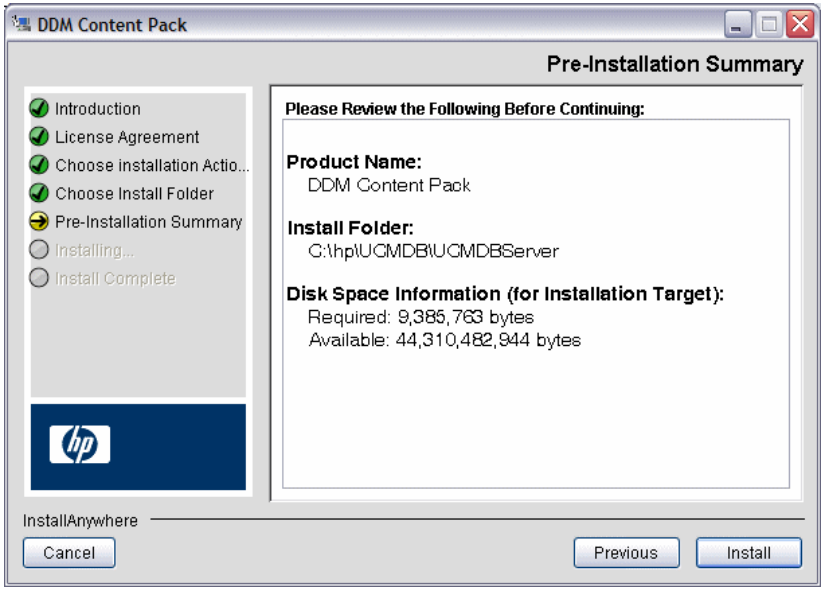


6 Accept the default entry or click **Choose** to display a standard Browse dialog box. Browse to and select the folder where the HP Universal CMDB server is installed.

Tip: To display the default installation folder again, click **Restore Default Folder**.

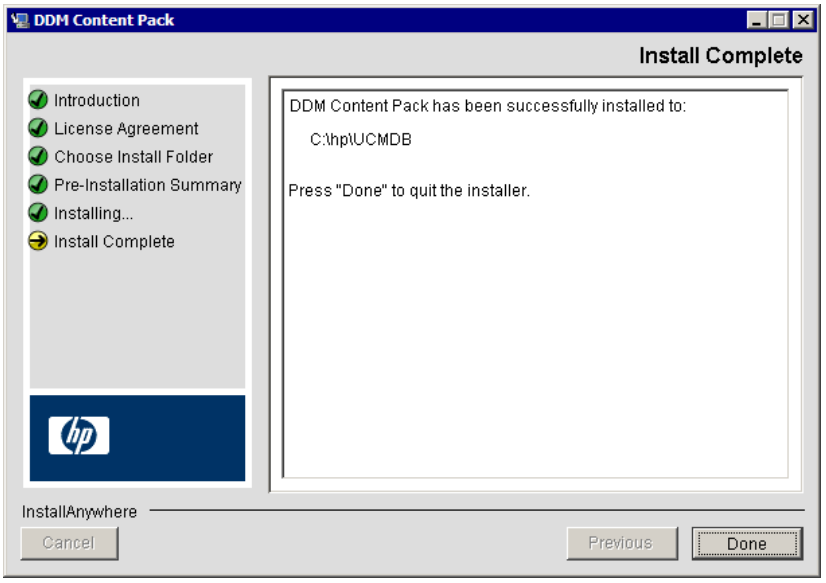
Click **Next**.

7 The Pre-Installation Summary dialog box opens and lists the installation options you have selected.



8 If you are satisfied with the summary, click **Install**. A message is displayed indicating that the installation is currently being performed.

The Install Complete dialog box opens:



Setup indicates the result of the installation:

➤ If installation is successful, click **Done**.

The installer copies all DDM packages to the **ddm_content_packages_3.0** and **factory_packages** folders under the following folder: **C:\hp\UCMDB\UCMDBServer\root\lib**.

This is the end of the installation procedure.

➤ If you chose not to deploy the DDM Content Pack 3.00 packages during installation, installation is successful, but you must now manually deploy the packages. Click **Done**, and continue to the procedure for deploying packages. For details, see "Deploy Packages" on page 20.

➤ If installation is not successful, that is, errors occurred during installation, a message is displayed. Refer to the installation log for details.

At the end of the Content Pack installation procedure, the following installation log file is created and placed under the UCMDB Server root folder: **DDM_Content_Pack_InstallLog.log**

Deploy Packages

If you did not enable automatic deployment of the DDM Content Pack packages (in step 5 on page 16), you must manually back up all DDM packages and deploy them to the UCMDB Server.

This task includes the following sections:

Back Up All DDM Packages

1 Launch the Web browser and enter the following address:

```
http://<machine name or IP address>.<domain_name>:8080/jmx-console
```

where **<machine name or IP address>** is the machine on which HP Universal CMDB is installed. You log in using the JMX console authentication credentials. For details, see "Working With the JMX Console" in the UCMDB *Reference Information* guide.

2 Click the **MAM** > **service=MAM Packaging Services** link.

3 In the JMX MBEAN View page, locate the following operation: **exportPackages ()**.

- In the **customerID** field, enter **1**.
- In the **packagesNames** field, leave this field empty to export all packages.
- In the **outputDir** field, enter the complete path to a directory where UCMDB should place the backed-up packages, for example, **C:\hp\UCMDB\UCMDBServer\root\lib\my_packages_backup**. This directory is automatically created.

  These packages are compatible with UCMDB version 8.01 or later.

  Tip: Keep this directory for future reference.
- In the **userOnly** field, select **False** to export all packages (and not only the user-created packages).

4 Click **Invoke**.

5 Verify that all relevant DDM packages have been backed-up to this folder, and that there are no errors in the **mam.packaging.log** file (located under the **j2f\log** directory).

Note: When installing CP 3.00 on top of HP UCMDB version 8.01, an exception may appear in the mam.packaging.log file, indicating that HP DDM job resources will not be overridden. These exceptions can be safely ignored. This defect has been fixed in HP UCMDB version 8.02, so the exceptions no longer appear.

Deploy DDM Content Pack Packages

6 In the same JMX MBEAN View page that you used to back up the DDM packages, locate the following operation: **deployPackages()**.

- In the **customerID** field, enter **1**.
- In the **dir** field, enter **C:\hp\UCMDB\UCMDBServer\root\lib\ddm_content_packages_3.0** (all DDM Content Pack packages are loaded from this folder).
- In the **packagesNames** field, leave this field empty to deploy all packages from this folder.
- In the **overrideCustomChanges** field, select **True** to override the DDM job configuration that was changed through the DDM application.

7 Click **Invoke**.

8 Verify that there are no errors in the **mam.packaging.log** file.

3

Supporting Multi-Lingual Locales

This chapter includes:

Concepts

Tasks

Reference

Overview

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 encodings: 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 encodings 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.

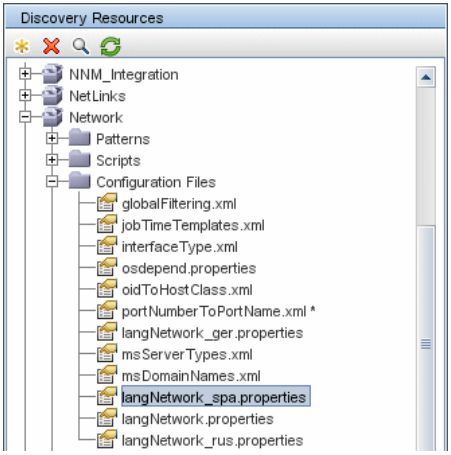
Once it is discovered in which language the command output is produced (# 1), possible character sets are limited to one or two (# 2). Furthermore, it is known which key words are contained in this output (# 5).

The solution, therefore, is to decode the command output with one of the possible encodings 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" in *Discovery and Dependency Mapping Guide*.



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 hardcoded, 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 28

➤ "Declare and Register the Language Object" on page 28

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 26.

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" in *Discovery and Dependency Mapping Guide*.

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 31. 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 HP Discovery and Dependency Mapping Web Service API" in *Discovery and Dependency Mapping Guide*.

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
```

To define a job:

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 Encodings 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.

Avoid Choosing Keyword When Command Is Executed

In certain use cases, it is enough to determine the encoding once and then to use it everywhere. 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 31. 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 27.

API Reference

This section includes:

The Language Class

This class encapsulates information about the language, such as resource bundle postfix, possible encodings, 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 33.

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.

4

SQL Server Discovery by OS Credentials

This chapter includes:

Concepts

Overview

This chapter describes how DDM discovers MS SQL Server CIs, using operating system (OS) credentials. DDM creates an identifiable SQL Server CI, rather than a generic Software Element CI.

Previously, SQL Server discovery assumed the existence of a process with the name of **sqlservr.exe**. Once DDM found this process, generic software elements with a **MSSQL DB** value in the **data_name** attribute were reported to UCMDB.

The current changes rectify the following issues:

- DDM did not discover instance name information, so identifiable CIs could not be reported.
- When several instances of SQL Server existed in the environment, DDM reported only one software element to UCMDB.

Discovery Jobs

The following jobs discover MS SQL Server components by OS credentials:

- Host Resources and Applications by Shell
- Host Resources and Applications by WMI

Discovery Changes

The following changes have been made:

- When running discovery with OS credentials, an identifiable SQL Server CI is reported, instead of the generic Software Element CI.
- DDM can now resolve the SQL Server instance name.
- The Probe can report multiple SQL Server instances, each of them linked by a **depend** link to its own **sqlservr.exe** process.
- DDM now supports SQL Server named instances.
- DDM supports discovery of the following SQL Server versions: MSSQL 2000, 2005, 2008.

Discovery by OS Credentials

There are two approaches to identifying MS SQL Server instance names by OS credentials. The changes appear in the Host_Resources_Basic package.

By Process Command Line

The SQL Server process usually includes the MS SQL Server instance name in its command line. DDM extracts this instance name to a CI.

Note: A process command line cannot be retrieved by the SNMP protocol. Therefore, SNMP cannot be used to discover the MS SQL Server instance name, and DDM reports the generic software element instead.

Using Windows Services

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

5

Class Model Enhancements

This chapter documents class model enhancements.

This chapter includes:

Concepts

Reference

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

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

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.

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.

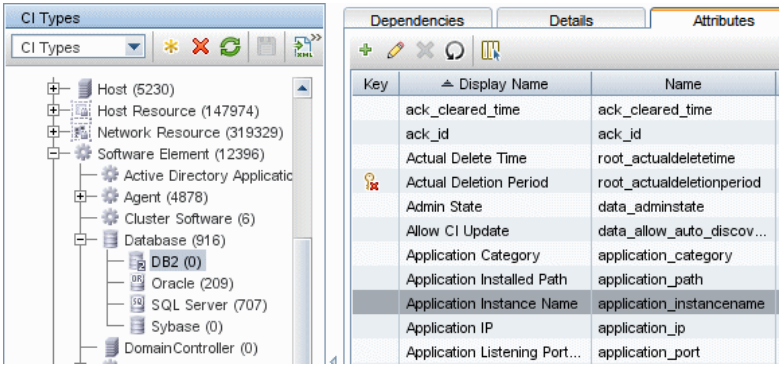
The regular expression for determining the product code (in Python notation):

```
"\\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.



Troubleshooting and Limitations

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

Index

R

S

U

W