HP OpenView Service Desk

Data Exchange Administrator's Guide

Software Version: 5.00 Patch 2

For the Windows and UNIX Operating Systems

Manufacturing Part Number: None
Document Release Date: August 2006
Software Release Date: August 2006

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.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

OpenView® is a registered U.S. trademark of Hewlett-Packard Company.

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

Contents

Contents

Contents

Contents

Documentation Updates

This manual's title page contains the following identifying information:

- 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, visit the following URL:

http://ovweb.external.hp.com/lpe/doc_serv/

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

Please visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software 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 valuable support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter 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 log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://www.managementsoftware.hp.com/passport-registration.html

1 Data Exchange

Data Exchange is the process of extracting information from a data source and importing it into another data source. The key steps involved in a data exchange process are as follows:

1. Create or modify a data extraction configuration file.
2. Run the data extractor to export the data to an XML file.
3. Import the data into the HP OpenView database.

After the data exchange process is configured for an external application, save the settings as a task, and use the task to export and import data again. See “Create a Data Exchange Task” on page 42 for more information.

NOTE You can start the import process from any system where the data exchange command line interface is available.

Related Topics

“Export Data” on page 18
“Import Mapping” on page 35
“Import Data” on page 41
“Reconcile Your Data” on page 58

Export Data

A number of extractors are available to export data. The purpose of the extractor is to pull data from where it is stored by one application and save it in a format that other applications can import.

Use a configuration file to set up the data export process. You can choose any number of external entities and properties to store in XML files. The XML format is used because it can be read and imported by many database applications without changing the extractor.

The data export process uses a data type definition file that is located in the same folder as the XML file. Activities performed by the data export process include the following:

- Validates the exported XML file.
- Checks the tag hierarchy and spelling.
- Determines if relations exist that reference objects in the exported XML file.
- Creates a log file.

Related Topics

"What to Export" on page 19
"Configure the Data Extractor" on page 20
"Export Data" on page 18

What to Export

Before you configure the data extractor, you need to know what information you want to export. Take the following steps to decide what to export:

1. Define the information to export.
2. Evaluate the structure of the external database to see how the information is stored. This evaluation should result in a list of table names, column names, and attributes. Use this list to map the ATT and COLUMNS attributes in the CLASSES section of the data extraction configuration file.
3. Determine if the external application's data source can provide the information or if you need to use an additional management application. Develop a list of external entity class names, attributes, key values, and relations to help you set up import mapping.

NOTE Loading only selected columns instead of complete tables can result in a much faster data exchange process.

Related Topics

"Configure the Data Extractor" on page 20
"Export Data" on page 18

Configure the Data Extractor

The extractor contains a number of sample configurations. These configuration files transmit commands to extract data from a defined external data source. Administrators use configuration files to specify the data to export as well as how and when to export the data.

Configuration file parameters are used to construct queries for the data source. The information received in response to these queries is exported in the defined format.

Configuration files are stored in the `data_exchange` directory.

To configure the Data Extractor, complete the following steps:

1. Create a new data extraction configuration file.
2. Define the external data source information, including type (ODBC, JDBC) and name (ODBC) or driver type (JDBC).
3. Specify the system settings. These settings include the location of log and output files, the name of the application from which you are exporting data, and the character encoding for the data extraction configuration file language.
4. Define the external entities classes to export. This includes the following:

   - Define class names. The class names you define represent the external entities you export and are used later to set up import mapping.

     The mapping you create depends on the type of information you export and import. For example, if you export personnel information, you could map it to the Person configuration item and the Internal Person template when you import it into HP OpenView.

   - Create and populate a CLASS section for each external entity. In each CLASS section, specify the SOURCE database from which to export the external entity data.
   - Define attributes in the ATT section for each external entity. The ATT section is used to identify exported objects and defines how to import them into HP OpenView.

- Define the COLUMNS you want to select in the external data source by specifying [columnname1] AS [alias 1]. If an alias is not specified, the column name will be used as the alias name.

  The COLUMN names you define must match the column names in the tables from which you want to extract data. In the case of a spread sheet, the names must match the names of the logical groups which contain the data to extract.

  You cannot use an external attribute more than once for import mapping. If you want to re use an external attribute, you have to read it twice in the COLUMNS section, for example:

  ```
  ATT=[IPADDRESS_A],[IPADDRESS_B]
  COLUMNS=[ADMINISTRATIVE].[IPADDRESS] as
  [IPADDRESS_A], [ADMINISTRATIVE].[IPADDRESS] as
  [IPADDRESS_B]
  ```

- Enter the CONDITION or selection criteria for the records to be retrieved.

Related Topics

"What to Export" on page 19
"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"SQL Statements in the Data Extraction Process" on page 32
"Export Data" on page 18

Data Extraction Configuration File

The following topics explain the keywords and syntax used in the various sections of the extractor configuration file.

Related Topics

"Data Extraction Configuration File Example" on page 30
"SQL Statements in the Data Extraction Process" on page 32
"Configure the Data Extractor" on page 20

Connection Section The Connection section defines the connection type to use for the data extraction process.

The Connection Keyword table shows the keyword you can use in the Connection section of the data extraction configuration file.

Table 1-1 Connection Keyword

Keyword	Description
TYPE	The default value is an ODBC connection. You can also select WIN32ODBC or LDAP. ODBC establishes a connection using the JDBC-ODBC bridge, which cannot connect to a WBEM data source. WIN32ODBC uses the native win32 ODBC driver.

Related Topics

"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"DSN Section" on page 24
"JDBC Section" on page 25
"System Section" on page 26
"Classes Section" on page 28
"Configure the Data Extractor" on page 20

DSN Section The DSN section is only required for ODBC connections in Microsoft Windows and defines the data source to be used for the extraction process.

Since ODBC is the default connection type, it does not need to be explicitly defined in the Connection section. If you specify a JDBC connection type, you need to define the details of the JDBC driver to use in the JDBC section of the configuration file.

The DSN Keywords for ODBC Links table shows the keywords used in the DSN section of the data extraction configuration file.

Table 1-2 DSN Keywords for ODBC Links

Keyword	Description
NAME	The name of the ODBC data source.
USR	The name of the user with access to the data source where the data to be exported resides.
PWD	The password of the user defined in USR.

Related Topics

"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"Connection Section" on page 23
"JDBC Section" on page 25
"System Section" on page 26
"Classes Section" on page 28
"Configure the Data Extractor" on page 20

JDBC Section The JDBC section is only required if TYPE=JDBC is specified in the Connection section. Values in this section differ according to whether you are using an Oracle or SQLServer data source.

The JDBC Keywords table shows the keywords used in the JDBC section of the data extraction configuration file.

Table 1-3 JDBC Keywords

Keyword	Description
DRIVER	The class name of the JDBC driver to be used, for example: • Oracle: `oracle.jdbc.driver.OracleDriver` • INET Driver for Microsoft SQL Server: `com.inet.tds.TdsDriver`
URL	The URL to connect to the database. For example: • Oracle (Thin): `jdbc:oracle:thin:@host:port:sid` • SQLServer: `jdbc:inetdae7:host:port`
USR	The database user who owns the data source tables and views
PWD	The password of the user defined in USR.

Related Topics

"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"Connection Section" on page 23
"DSN Section" on page 24
"System Section" on page 26
"Classes Section" on page 28
"Configure the Data Extractor" on page 20

System Section The System section defines the system and settings for the files written by the data extraction process.

The System Keywords table shows the keywords used in the System section of the data extraction configuration file.

Table 1-4 System Keywords

Keyword	Description
LOG	Specify TRUE/FALSE to create/not create log file.
XML	Specify TRUE/FALSE to create/not create an XML file.
LOG_FILE	The location and name of the log file. This field is only used when you run a data exchange task from a command line. Spaces are allowed only if the complete path/file name is enclosed in quotes, for example: `C:\Program Files\HP OpenView\log`
XML_OUTPUT_FILE	The location and name of the extracted XML file. This field is only used when you run Data Exchange Tasks from a command line. Spaces are allowed only if the complete path/file name is enclosed in quotes. If CLASS_TO_XML=TRUE, the file specified in XML_OUTPUT_FILE is not written. For more information, see CLASS_TO_XML.
APPLICATION_NAME	The name of the external application you are extracting data from.
ENCODING	The name of the character encoding you want to use when viewing the XML file. UTF-8 is the default.

Table 1-4 System Keywords (Continued)

Keyword	Description
SQL_TO_UPPERCASE	Default is TRUE. TRUE converts all SQL queries to uppercase and is required for ORACLE databases. FALSE allows both upper and lower case SQL queries and is required by some other databases.
CLASS_TO_XML	Assume FALSE if no value is entered. If set to TRUE, the file specified in XML_OUTPUT_FILE is not written and each parent class in the Classes section is extracted to a separate XML file. The XML files are stored in the directory defined in XML_OUTPUT_FILE.

Related Topics

"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"Connection Section" on page 23
"DSN Section" on page 24
"JDBC Section" on page 25
"Classes Section" on page 28
"Configure the Data Extractor" on page 20

Classes Section Classes define the objects you want to extract. These objects are specified in more detail in the Classes.

The Classes Keywords table shows the keywords used in the Classes section of the data extraction configuration file.

Table 1-5 Classes Keywords

Keyword	Description
NAME	The name you specify for the object you want to import. This is the class name that is mapped from the XML file to an object when you do the import mapping.
SOURCE	The name you specify for the data source containing the tables or views from which you select data.
ATT	Defines the contents of the classes and aliases for the attributes you want to export. Attributes can be the same as COLUMNS or aliases and will be captured from the columns defined in the COLUMNS section.
COLUMNS	Specifies all columns to be selected with an alias, [columnname1] AS [alias 1]. Not all columns specified in COLUMNS must appear in the XML file. Columns defined in PARENT_RELATION must be put into the column list when tables are cached. The extractor uses this information to find all children for a particular parent in memory.
MAXRECORDS	Specifies the maximum number of records to be exported.
LOADTABLE	Specify TRUE/FALSE to cache/not cache records in memory. Caching means parent child relations process more quickly. Choose TRUE if you have plenty of memory and want quicker performance.

Table 1-5 Classes Keywords (Continued)

Keyword	Description
PARENT	Specifies the parent class.
PARENT_RELATION	Specifies the relation between the child and parent. The value preceding the equals sign is the foreign key used to identify the COLUMN of the child. The value after the equals sign is the alias of the primary key of the parent.
PARENT_RELATION_NAME	Specifies the name of the relation. The default is PARENT.
CONDITION	Provide conditions or selection criteria for the records to be retrieved.
ORDERBY	Specifies the order of the records.

Related Topics

"Data Extraction Configuration File" on page 22
"Data Extraction Configuration File Example" on page 30
"Connection Section" on page 23
"DSN Section" on page 24
"JDBC Section" on page 25
"System Section" on page 26
"Configure the Data Extractor" on page 20

Data Extraction Configuration File Example

The following example shows an excerpt from a data extraction configuration file for an ODBC link. The DSN section at the start of the excerpt is required only for ODBC connections in Windows operation systems. A JDBC connection changes the connection type to TYPE=JDBC and replaces the DSN section with its own JDBC section.

The data source NAME in the DSN section of the data extraction configuration file is the name you specify when you create an ODBC link. Each data extraction configuration file for an ODBC link must have a unique data source name.

Example 1-1 Data Extractor Configuration File Example

```
[DSN]
NAME=hpdtadb_dta
USR=hpdta
PWD=hpdta

[SYSTEM]
LOG=TRUE
XML=TRUE
LOG_FILE=C:\Temp\DTA5.log
XML_OUTPUT_FILE=C:\Temp\DTA5.xml
APPLICATION_NAME=DTA5
ENCODING=UTF-8
SQL_TO_UPPERCASE=TRUE

[CLASSES]
NAME=ADMIN,CPU

[ADMIN]
SOURCE=ADMINISTRATIVE
ID=[MACHID]
ATT=[MACHINENAME],[IPADDRESS]
COLUMNS=[ADMINISTRATIVE].[MACHINEID] as
[MACHID],[ADMINISTRATIVE].[MACHINENAME] as
[MACHINENAME],[ADMINISTRATIVE].[IPADDRESS] as [IPADDRESS]
CONDITION=[OSTYPE]='Windows NT' OR [OSTYPE]='Windows 98'

[CPU]
SOURCE=CPU
PARENT=ADMIN
```

```
PARENT_RELATION=[CPU].[MACHINEID]=MACHID
ID=[CLOCKSPEED],[CPUTYPE]
COLUMNS=[CPU].[MACHINEID] as [MACHID],[CPU].[CLOCKSPEED] as
[CLOCKSPEED],[CPU].[CPUTYPE] as [CPUTYPE],[CPU].[REPLICATE]
as [REPLICATE]
```

The data exchange process uses the "ID" tag internally to give each record a unique identity in the XML file. For this reason, the data extractor configuration file can contain only one "ID" tag. To prevent the occurrence of multiple “ID” tags, make sure that your data extractor configuration file does not contain any explicit “ID” values in the ATT and COLUMNS sections.

The following example shows you how you can use an alias to ensure that “ID” in the data extraction configuration file is understood as the value defined in IDENTIFIER and, as a result, does not interfere with the data extractor’s internal use of the “ID” tag.

Example 1-2 Avoid Multiple ID Tags in ATT and COLUMNS

```
[domain]
SOURCE=domain
ATT=[IDENTIFIER],[NAME],[SEARCHCODE]
COLUMNS=[ID] as [IDENTIFIER],[NAME],[SEARCHCODE]
LOADTABLE=TRUE
```

Related Topics

“Data Extraction Configuration File” on page 22
“SQL Statements in the Data Extraction Process” on page 32
“Configure the Data Extractor” on page 20

SQL Statements in the Data Extraction Process

SQL statements are generated by class definitions in the data extraction configuration file. These statements are passed to the ODBC driver. Tables, columns, and classes are selected according to the definitions in the configuration file.

The extractor processes SQL statements as follows:

1. For each class without parents, create an SQL statement. Each record is extracted and placed in the data exchange file individually.
2. For each child class,

   if LOADTABLE=TRUE or if no records are loaded, create an SQL statement to load all records in memory, and use the PARENT_RELATION to scan through the records in memory to find the children;

   else,

   create an SQL statement to get the records belonging to each parent with the PARENT_RELATION. Process all child records recursively
3. The following records are created from the class section:

   - **LOADTABLE=TRUE** or no child class:

     ```
     SELECT <COLUMNS>
     FROM <SOURCE>
     WHERE <CONDITON>
     ORDER BY <ORDERBY>
     ```

   - **LOADTABLE=FALSE** and it is a child class **PARENT=**:

     ```
     SELECT <COLUMNS>
     FROM <SOURCE>
     WHERE <CONDITON>AND<PARENT_RELATION>
     ORDER BY <ORDERBY>;
     ```

   - joins (LEFT JOIN...)

     ```
     SELECT <COLUMNS>
     FROM <SOURCE>ON<CONDITION>
     ORDER BY <ORDERBY>
     ```

Related Topics

"Data Extraction Configuration File" on page 22

"Data Extraction Configuration File Example" on page 30
"Configure the Data Extractor" on page 20

Import Mapping

Import mapping enables you to define how data from an external data source appears after being imported into the target database. You can extract all or part of the data available.

When you map classes in an external data source to classes in the target database, you ensure data integrity. If the default attributes and values defined in the templates provided do not meet the requirements of your environment, you can create new fields to support the mapping of additional values.

For an overview of the objects whose attributes you need to map for import, see the “Classes Section” on page 28.

NOTE External classes and attributes used for import mapping must match those present in the XML file. Object names are case sensitive. If an object name is incorrect, a warning is entered in the import log file during the import process.

The order of the classes in the data extraction configuration file determines the order for import mapping. Ideally, you should create the entire data extraction configuration file before you set up import mapping. This practice allows you to base the import mapping on the information in the XML file.

Related Topics

“Entity Mapping” on page 36
“Data Exchange” on page 15

Entity Mapping

Every external class that you import must be mapped to an internal object. You can use one of the templates provided or create a new template. Templates often contain default values for attributes and provide a way of relating an external class to an internal object.

When you want to import data, look through the available templates and select the template that comes closest to matching the type of data you want to import. A template may perfectly meet the needs of your environment, or you may need to modify the default attributes and values.

Classes and their associated properties must be mapped to be imported. When classes are mapped, the import process knows how to format the data and where to store it. The class names defined in the CLASSES section of the data extraction configuration file are the classes you map in the import mapping process. See “Classes Section” on page 28 for more information.

Related Topics

“Attribute Mapping” on page 37
“Import Mapping” on page 35

Attribute Mapping

Attributes define and identify objects. For example, an exported class has the properties Model and Location, and an internal application object has similar attributes called Type and Address. For the properties Model and Location to be correctly extracted, exported, and imported, they must be mapped to the attributes Type and Address.

In the data extraction configuration file, the ATT and PARENT_RELATION entries for each configured class determine the attributes that must be considered in import mapping. The attributes in the ATT line of the data extraction configuration file are mapped to fields that are available in the template you specify when setting up the import mapping.

Related Topics

"Key Binding" on page 38
"Attribute Values" on page 39
"Entity Mapping" on page 36
"Import Mapping" on page 35

Key Binding Key binding is used to identify a specific instance of an object. By setting the key binding for an attribute, you can identify similar objects and recognize changes.

At least one external attribute in every class must be used for key binding. The **Name** property is often set as a default key value.

Related Topics

"Attribute Values" on page 39
"Attribute Mapping" on page 37
"Entity Mapping" on page 36
"Import Mapping" on page 35

Attribute Values Some attributes contain values in a code table. These values are a predefined set of possible values to choose from. When you map an attribute that has values in a code table, the value mapping button is available in the **Attribute Mapping** dialog box to assist you.

When required fields exist for an object that you want to import, that object must be mapped. See “Required Fields” on page 69 for more information.

Related Topics

“Key Binding” on page 38
“Attribute Mapping” on page 37
“Entity Mapping” on page 36
“Import Mapping” on page 35

Import Data

You can import data from the console, or you can use the `OvObsImporter` command from a command line. See OvObsImporter(1m) for more information.

If the unique key of a record is changed, the data exchange process will treat it as a new object for import. If the unique key is not changed, the data exchange will update the record and overwrite the existing value.

Old objects and relations must be deleted manually.

Related Topics

"Create a Data Exchange Task" on page 42
"Start a Data Exchange Task" on page 43
"Schedule a Data Exchange Task" on page 44
"Create a Data Exchange Task Group" on page 45
"Scalable Importing" on page 47
"Data Exchange" on page 15

Create a Data Exchange Task

You can set up multiple tasks for exporting and importing data.

To create a Data Exchange Task, complete the following steps:

1. Log on the HP OpenView console using a Role that includes administrator privileges.
2. Select the **HP OpenView Configuration** workspace.
3. In the **Data** workspace, select **Data→Data Exchange→Data Exchange Task**.
4. Right-click in the right pane and select **New Data Exchange Task** from the drop-down menu. The Data Exchange Task dialog box displays.
5. Select the **Export** check box if you want to export data. In the **Export Configuration** field, enter the name of the data extraction configuration file you want the extractor to use to export the data.
6. In the **Exchange File** field, enter the name of the exported data exchange file that you want to import after the export process has completed.

   The name of the data exchange file is set in the extractor's configuration file.
7. Select the **Import** check box if you want to import data following the Export. Provide **Account** and **Import Mapping** information.
8. If you want to start the task immediately, click **Action→Start**.
9. Save your task and close the dialog box.

Related Topics

"Start a Data Exchange Task" on page 43
"Schedule a Data Exchange Task" on page 44
"Create a Data Exchange Task Group" on page 45
"Scalable Importing" on page 47
"Import Data" on page 41

Start a Data Exchange Task

To start a Data Exchange Task, complete the following steps:

1. Log on the HP OpenView console using a Role that includes administrator privileges.
2. Select the **HP OpenView Configuration** workspace.
3. Select the **Data** workspace.
4. In the **Data** workspace, select **Data→Data Exchange→Data Exchange Task**.
5. Right-click a task in the right pane and select **Start** from the drop-down menu.

Related Topics

"Create a Data Exchange Task" on page 42
"Schedule a Data Exchange Task" on page 44
"Create a Data Exchange Task Group" on page 45
"Scalable Importing" on page 47
"Import Data" on page 41

Schedule a Data Exchange Task

You can use the console or the command line to schedule Data Exchange Tasks to run automatically at a specified time.

On Windows operating systems, you can use the `at` command to specify the date and time to run a Data Exchange Task. Alternatively, you can schedule a task using the Control Panel.

NOTE The Schedule service must be running on the system where you want the task performed.

On UNIX and Linux operating systems, add an entry to a user's `crontab`, which runs a command or a script at a scheduled time.

NOTE The user starting the `cron` job must have appropriate access to the locations specified in the paths to the various configuration and log files.

Related Topics

"Create a Data Exchange Task" on page 42
"Start a Data Exchange Task" on page 43
"Create a Data Exchange Task Group" on page 45
"Scalable Importing" on page 47
"Import Data" on page 41

Create a Data Exchange Task Group

You can relate multiple Data Exchange Tasks and start them as a group of Data Exchange Tasks in a specified order.

NOTE You cannot start a Data Exchange Task Group from the command line.

To create a Data Exchange Task Group, complete the following steps:

1. Log on the HP OpenView console using a Role that includes administrator privileges.
2. Select the **HP OpenView Configuration** workspace.
3. Select the **Data** workspace.
4. In the **Data** workspace, select **Data→Data Exchange→Task Groups**.
5. Right-click in the **Task Groups** pane and select **New Data Exchange Task Group** from the drop-down menu.
6. Enter a short description to identify the task group in the **Description** field.
7. Click **Relate** to locate and add tasks to the task group.
8. Tasks are started according to the order in which they appear in the **Data Exchange Task Group**. Use the **Preview** button to display basic details of individual tasks.

   Use the **Move Up** or **Move Down** buttons to change the order in which the tasks are run.
9. Save your task group and close the dialog box.

NOTE To remove a task from a task group, select the task in the group window and click **Unrelate**. Then click **File→Save** to save the changes.

Related Topics

"Create a Data Exchange Task" on page 42
"Start a Data Exchange Task" on page 43
"Schedule a Data Exchange Task" on page 44

“Scalable Importing” on page 47
“Import Data” on page 41

Scalable Importing

Scalable importing is the process of configuring and using multiple systems to collectively import unusually large amounts of data into the HP OpenView database. You can also configure systems to commit data to the database with multiple application servers.

The operation works by breaking the data down into smaller chunks and using a collection of client systems to process the data chunks. Scalable importing is only recommended for large import operations, which would normally take several hours to complete.

NOTE

Scalable importing is sometimes known and referred to as parallel importing. On the command line, you use the `OvObsImporter` command with the `-parallel` option to specify the shared directory to be used to process data during a scalable import. See OvObsImporter(1m) for more information.

In a scalable import operation, one system is assigned to be the Master, and a collection of systems are the Servants. The Master manages the task of allocating data chunks. The Servants process the data chunks and pass them on to the assigned application server. When the Master completes the task of creating data chunks, it takes on the role of a Servant to help process the data chunks.

Related Topics

“Prepare the Environment” on page 48
“Export the XML Files” on page 50
“Perform a Scalable Import” on page 51
“Perform a Scalable Import as a Task Group” on page 52
“Review the Log Files” on page 54
“Troubleshoot Scalable Importing Errors” on page 55
“Import Data” on page 41

Prepare the Environment

You need a specific environment to perform a scalable import.

To set up an environment for scalable importing, complete the following steps:

1. Assign one system the role of Master. The Master system should have the greatest processing power.
2. Make a directory available for processing the chunks of data. Share or export the directory to be used by the Master and Servants. The Master and Servants need read/write access.

NOTE

For performance reasons, it is recommended that you set up the shared directory on the Master system.

Set the number of connections to the maximum allowed so that Servants do not time out waiting to connect.

Master and Servants write log files to a shared data directory: `<SharedDataDir>\errors\`. See “Review the Log Files” on page 54 for more information.

3. Assign multiple systems the role of a Servant. You need at least twice as much combined processing speed in the Servants as you have in the single Master.

   If you are using more than 20 Servants, you may need to increase the polling interval of each Servant. The correct polling interval helps prevent the Master from spending too much time handling requests. To determine the optimum polling interval for a Servant, multiply the number of Servants by 500 milliseconds.
4. Link each Servant to the data processing directory. Map or mount a directory on each Servant to the shared data processing directory containing the imported data.
5. Create an integration account for the import process. This account information is required when you start the scalable import process on both the Master and the Servants.

Related Topics

“Export the XML Files” on page 50

Export the XML Files

The XML files used for scalable importing must be created with the `OvObsExporter` command. See OvObsExporter(1m) for more information.

The configuration files used for the export must be configured to export single, not multiple references. In some cases, you will need to split an existing data extraction configuration file into several, separate data extraction configuration files.

NOTE

In the [SYSTEM] section of your data extraction configuration file, you can add the line, CLASS_TO_XML=TRUE to create an XML file for each parent class that is exported.

Related Topics

"Prepare the Environment" on page 48
"Perform a Scalable Import" on page 51
"Perform a Scalable Import as a Task Group" on page 52
"Review the Log Files" on page 54
"Troubleshoot Scalable Importing Errors" on page 55
"Import Data" on page 41

Perform a Scalable Import

To perform a scalable import, use a command to start the scalable import process on each Servant system. When all the Servants are running, start the Master, which will then make data available to the Servants for processing.

Use the `OvObsImporter` command to start the scalable import. See OvObsImporter(1m) for more information.

Related Topics

"Prepare the Environment" on page 48
"Export the XML Files" on page 50
"Perform a Scalable Import as a Task Group" on page 52
"Review the Log Files" on page 54
"Troubleshoot Scalable Importing Errors" on page 55
"Import Data" on page 41

Perform a Scalable Import as a Task Group

It is possible to use one file to execute a group of tasks from a Master system at a scheduled time.

The script or batch file you write should list the sequence of `OvObsImporter` commands you want to use to start the individual scalable import tasks. See OvObsImporter(1m) for more information.

The example, "Starting Scalable Importing from a Script on the Master", assumes that the configured Servants are already running, and the shared data directory is on the Master system where the batch file is to run.

If the shared data directory is somewhere else on the network, replace `<SharedDataDir>` with the mapped drive or mounted directory pointing to the shared data directory.

Example 1-3 Starting Scalable Importing from a Script on the Master

```
OvObsImporter imp_account Italy macaroni \
-data=user_accounts_1.xml -mapping=user_account_mapping \
-parallel=<SharedDataDir> -form -close \
-logfile=user_accounts1.log -xmlsize=250

OvObsImporter imp_account Italy macaroni \
-data=user_accounts_2.xml -mapping=user_account_mapping \
-parallel=<SharedDataDir> -form -close \
-logfile=user_accounts2.log -xmlsize=250

OvObsImporter imp_account Italy macaroni \
-data=user_accounts_3.xml -mapping=user_account_mapping \
-parallel=<SharedDataDir> -form -close \
-logfile=user_accounts3.log -xmlsize=250

OvObsImporter imp_account Italy macaroni \
-data=user_accounts_4.xml -mapping=user_account_mapping \
-parallel=<SharedDataDir> -form -close -stopservants \
-logfile=user_accounts4.log -xmlsize=250
```

Related Topics

"Prepare the Environment" on page 48
"Export the XML Files" on page 50
"Perform a Scalable Import" on page 51
"Review the Log Files" on page 54
"Troubleshoot Scalable Importing Errors" on page 55

“Import Data” on page 41

Review the Log Files

The Master and each Servant create a log and error log for each data exchange file that is processed. The logs are saved in a dedicated directory named `log`, which resides under the shared data processing directory that you set up for the scalable import operation.

- For Windows operating systems:
  `\\<HostName>\<SharedDataDir>\log\<ServantHostName>\<DataExchangeFileName>.log`
- For UNIX and LINUX operating systems:
  `/<SharedDataDir>/log/<ServantHostName>/<DataExchangeFileName>.log`

The name of the log file is defined with the `-logfile=` command option. The name of the sub directory in which the logs reside is determined by the host name of the system, which writes the log. This means that you can quickly determine which Servant was responsible for which logs.

Duplicate log file names are not allowed. If a log file of the same name already exists, the next log file name adds an extra number to the file name.

Servants record errors encountered during the processing of a data exchange file to an error log file. Error log file names are defined as follows:

`<DataExchangeFilename>_error.log`

Related Topics

"Prepare the Environment" on page 48
"Export the XML Files" on page 50
"Perform a Scalable Import" on page 51
"Perform a Scalable Import as a Task Group" on page 52
"Troubleshoot Scalable Importing Errors" on page 55
"Import Data" on page 41

Troubleshoot Scalable Importing Errors

The log and error log files can help you troubleshoot the cause of problems encountered by Servants when processing data during the scalable import operation.

Data chunks that cause an entry in the error file are placed in a dedicated directory named `errors`, which is a sub directory of the shared data directory that you set up for the scalable import operation. The name and location of these data chunks is not configurable.

Two types of errors may occur during the scalable import process:

- Time Out

  After the Master breaks down the data exchange file into smaller chunks, it waits for a Servant to indicate that it has finished processing the work. When a Servant finishes processing a data chunk, the indication of success is that the chunk is deleted.

  The Servant time out is set by default to 20 minutes. If a Servant takes longer than 20 minutes to process a chunk of data, the Master takes the original XML chunk and places it in a dedicated error folder with the file name prefix `time_out`.

- Fatal Error:

  If a Servant encounters a severe error when importing a data chunk, it stops the data processing task, and the data chunk it was working on is placed in the error folder with the file name prefix `fatal_error`. When the file is moved, the file lock on the data chunk is removed.

The name of the data chunks placed in the `error` directory is determined by the following file naming convention:

`<ErrorType>_<DataChunkFileName>.xml`

If a time out or fatal error occurs, the following steps my be helpful in locating and solving errors:

1. Search the error log for the problem that caused the error.
2. If possible, open the XML data chunk that is in the error folder, repair the problem that caused the error, and try importing it again. If the error is too severe, the entire scalable import task will have to be performed again.

3. If you cannot find a relevant entry in the error log or any further indication of what went wrong, the XML file may be corrupt. If the XML file opens and it looks correct, try importing it again. If it cannot be opened, you will need to import the entire data exchange file again.

To reimport a data chunk that previously caused an error, copy the appropriate data chunk from the error folder back to the shared data directory, and start the `OvObsImporter` process again, this time without using the `-close` parameter. See OvObsImporter(1m) for more information.

For example, if data chunk number 12 produced an error during the import of the data exchange file `user_accounts_1.xml`, you would enter the following command to try reimporting it:

```
OvObsImporter imp_account Italy localhost \
-data=fatal_error_user accounts_1_12.xml \
-mapping=user_account_mapping -parallel=<SharedDataDir> \
-form -logfile=user_accounts1_12
```

Check the import progress dialog for warnings and errors during the import process. If no errors or warnings occur, the import was successful.

Related Topics

“Prepare the Environment” on page 48
“Export the XML Files” on page 50
“Perform a Scalable Import” on page 51
“Perform a Scalable Import as a Task Group” on page 52
“Review the Log Files” on page 54
“Import Data” on page 41

Reconcile Your Data

The data reconciliation feature is called delta processing. It compares the current and previous data exchange files to find changes and identify duplicate information.

Delta processing creates a list of obsolete objects that require manual removal and sorts the data for import. By filtering redundant objects, attributes, and relations, you minimize the amount of data you have to import. If you have already imported data from a data source to HP OpenView, you can use the data reconciliation option to reduce the load on your system resources by ensuring that only the changes in your data are imported.

Table 1-6 on page 58 explains how the data reconciliation process determines which objects to import.

Table 1-6 Import Objects with Data Reconciliation

Object in previous exchange file?	Object in current exchange file?	Changes to the object's attributes?	How is object processed?
No	Yes	Not Applicable	Import object
Yes	No	Not Applicable	Remove object manually
Yes	Yes	No	No action taken
Yes	Yes	Yes	Import object = update attributes

Table 1-7 on page 59 explains how the data reconciliation process determines which relations to import.

Table 1-7 Import Relations with Data Reconciliation

Object in previous exchange file?	Object in current exchange file?	Is the relation new or changed?	Has the relation been removed?	How will relation be processed.
No	Yes	Yes	Not Applicable	Import relation
Yes	No	Not Applicable	Yes	Remove relation manually
Yes	Yes	No	No	No action taken
Yes	Yes	Yes	No	Import relation
Yes	Yes	No	Yes	Remove relation manually

Related Topics

"Delta Processing" on page 60
"Data Exchange" on page 15

Delta Processing

You can set up data reconciliation two ways:

- In the console select the **Use Delta Processing** option in the **Import** section of the **Data Exchange Task** dialog.
- On a command line, use the `-recofile=` option with the `OvObsImporter` command. See OvObsImporter(1m) for more information.

Delta processing compares the data collected in the current data exchange process with the contents of a previous data exchange operation. Differences and duplications are listed in a change log.

If you use the delta processing option, new objects and relations that are discovered during the reconciliation process are automatically imported. Any modified objects and relations are automatically updated.

Related Topics

"Remove Redundant Objects and Relations" on page 61
"Data Exchange" on page 15

Remove Redundant Objects and Relations

The data reconciliation process creates a change log, which records all new, modified, and redundant objects and relations. By using the key fields and values associated with each entry in the change log, you can locate obsolete or redundant objects quickly and easily.

NOTE To complete the reconciliation process, you need to remove obsolete objects and relations manually before you start the import process.

The data reconciliation process creates the change log in the `<OvDataDir>\conf\obs\data_exchange\log` directory and indicates the date and time at which the log file was created, for example; `DTA_Changes_200101101644.log`.

The values of all key fields are cached during import to make it possible to locate parent objects and relations when processing a relation. The smallest memory format practical is used.

Related Topics

"Delta Processing" on page 60
"Reconcile Your Data" on page 58

2 Import Data from External Events

The `OvObsLoadObject` command is an integration tool that external applications can use to interact with HP OpenView without user

intervention. The `OvObsLoadObject` command is designed to integrate external applications and use the information provided to keep the HP OpenView database automatically up to date. See OvObsLoadObject(1) for more information.

Related Topics

"Manage Event Queues" on page 65

Manage Event Queues

When many external events occur at or around the same time, a large number of entries can appear in event queues. When this happens, the order in which entries are processed can become a problem. If multiple entries contain data relating to the same external event, the entries must be processed to avoid errors when the data is committed to the database.

NOTE

The event queue tools described in this section are not designed to handle large amounts of events due to event storms in external applications. You need to correlate events on the external system to reduce the amount of data sent to HP OpenView.

Queue entries are not processed at the same speed. The processing speed depends on the contents of the queue entry and the load on the system at processing time. In extreme cases, the management server could still be processing a request to insert an object when another request arrives to update that object. In this case, the update request would fail since the object to be updated does not yet exist.

Queuing tools ensure that queue entries are processed one at a time and in the order that they were created.

Related Topics

"Create and Populate Queues" on page 65
"Import Data from External Events" on page 63

Create and Populate Queues

The following tools and scripts are provided for the purpose of event queue management:

- The "enqueue Tool" on page 66 adds new entries to an existing queue.
- The "dequeue Tool" on page 66 removes entries from a queue one by one and executes the command defined in each entry.

- The “queuectl Tool” on page 67 manages a queue and allows you to block, pause, and flush queues. You can also report the status of a queue.
- The “addentry Tool” on page 68 adds entries to a queue and starts the dequeue tool as necessary.

NOTE The version of this script for MS Windows operating systems is called `addentry.cmd`.

Related Topics

“Manage Event Queues” on page 65
“Import Data from External Events” on page 63

enqueue Tool

The `enqueue` tool creates a new entry in the event queue. The command takes the following form:

```
enqueue <queuename> <command>
```

The queue directory must exist prior to this action.

Related Topics

“dequeue Tool” on page 66
“queuectl Tool” on page 67
“addentry Tool” on page 68
“Create and Populate Queues” on page 65

dequeue Tool

The `dequeue` tool processes entries in a queue by reading and executing them one by one. The command takes the following form:

```
dequeue <queue_name>
```

After the `dequeue` tool successfully executes an entry, it deletes the entry from the queue. The dequeue tool exits when all entries in the queue are executed or when the queue is blocked.

Related Topics

“enqueue Tool” on page 66
“queuectl Tool” on page 67

"addentry Tool" on page 68
"Create and Populate Queues" on page 65

queuectl Tool

The `queuectl` tool allows you to manage and control access to queues. The command takes the following form:

```
queuectl <queuename> <option>
```

Table 2-1, "queuectl Tool Options," lists the options you can use with the `queuectl` command.

Table 2-1 queuectl Tool Options

BLOCK	Block all access to a queue and stop any running dequeue process. No new entries can be added.
FLUSH	Empty a queue without executing the entries.
UNBLOCK	Remove block from a queue. Entries can be added, and dequeue can be restarted.
ENTRIES	The number of entries in a queue.
STATUS	The current status of a queue. Possible status return codes are: • IDLE (exit code 10) No `dequeue` command is running on this queue. • RUNNING (exit code 11) The `dequeue` command is processing this queue. • BLOCKED (exit code 12) Queue is blocked by `queuectl` command. • HUH (exit code 13) Specified queue is in an unknown state.

Related Topics

"enqueue Tool" on page 66
"dequeue Tool" on page 66
"addentry Tool" on page 68
"Create and Populate Queues" on page 65

addentry Tool

The `addentry` tool does the following:

- Add entries to a queue with the `enqueue` command.
- Requests the status of a specified queue with the `queuectl` command.
- Starts the `dequeue` tool if necessary to run the commands defined in the queue entries.

The `addentry` script requires the following usage:

- For MS Windows operating systems: `addentry.cmd <queuename><flag>`
- For UNIX and Linux operating systems: `addentry.sh <queuename><flag>`

Related Topics

"enqueue Tool" on page 66
"dequeue Tool" on page 66
"queuectl Tool" on page 67
"Create and Populate Queues" on page 65

3 Data Exchange Commands

This chapter provides reference information for the following data exchange commands.

OvObsExporter(1m)

NAME

OvObsExporter – extract data from an external data source and store it in an XML file.

SYNOPSIS

```
OvObsExporter -h |help

OvObsExporter -f ConfigFile [-x OutputXmlFile | -l
OutputLogFile | -s(ilent)]
```

DESCRIPTION

`OvObsExporter` allows you to extract data from an external datasource defined in a configuration file and store the data in an XML file, which conforms to CIM/V2 DTD. You can use the `OvObsExporter` command to manage migrations from one database to another, or upgrade an existing database with data from another. The `OvObsExporter` command can read from a number of datasources, such as; generic ODBC, Win32 ODBC, JDBC, and LDAP.

Parameters

The `OvObsExporter` command recognizes the following parameters and options:

`-h |help`

Displays the parameters and options described in this list along with a short explanation

`-f <ConfigFile>`

The name of the file storing the export configuration. This parameter is *mandatory*: the configuration defines the nature and scope of the export operation, for example: the datasource name, precisely what data is to be exported, and how and where the export should take place.

Note that spaces in a file name or path are not allowed unless the complete path is enclosed in quotes, for example: `"C:\Program Files\HP Openview\cfgfilename.xml"`

`-x <OutputXmlFile>`

The name of the XML file, to which you want to write the output generated by the export operation. The format of the generated XML file conforms to the CIM/V2 DTD.

Note that spaces in a file name or path are not allowed unless the complete path is enclosed in quotes, for example: `"C:\Program Files\HP Openview\filename.xml"`

`-l <OutputLogFile>`

The name of the log file, to which you want to write any information and warning messages generated during the export operation.

Note that spaces in a file name or path are not allowed unless the complete path is enclosed in quotes, for example: `"C:\Program Files\HP Openview\filename.log"`

`-s(ilent)`

Disable the progress bar during the export operation. In silent mode, there is no indication of any status until the export operation completes.

AUTHOR

`OvObsExporter` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show how to use the `OvObsExporter` command and some of its options to control the import of data into the management-server database.

- To export data to the file `ObsExport.xml` using configuration details in the file `/opt/OV/data/data_exchange/config/incidents.ini` and writing progress information to the logfile `ObsExport.log`:

  OvObsExporter -f /opt/OV/data/data_exchange/config/incidents.ini -x ObsExport.xml -l ObsExport.log

SEE ALSO

OvObsImporter(1m)

COPYRIGHT

OvObsImporter(1m)

NAME

OvObsImporter – import data from a file into a data source.

SYNOPSIS

```
OvObsImporter -h |help

OvObsImporter -recohelp

OvObsImporter -parallelhelp

OvObsImporter <username> <password> <server>
-data=<XmlFilename> -mapping=<mapping>
[source[-xmlsize=<Kb>] | [-debug] |
[-logfile=<filename>] | [-tempdir=<directory>] | [-form]

OvObsImporter <username> <password> <server>
-data=<XmlFilename> -mapping=<mapping>
-recofile=<filename> [-changefile=<filename>]
[-sortsize=<Mb>] [-norecosort] [-nodatasort]
[-notimestamp] [-keepsortedfiles] [-norecoimport]
[-debug] [-logfile=<filename>][-tempdir=<directory>]
[-form]

OvObsImporter <username> <password> <server>
-data=<XmlFilename> -mapping=<mapping> -parallel=<drive>
[-polltime=<millisecs>] [-xmlsize=<Kb>]
[-servanttimeout=<seconds>]
[-stopservants][-logfile=<filename>][-tempdir=<directory
>] [-form] [-debug]
```

DESCRIPTION

`OvObsImporter` allows you to import data stored in an XML file (which conforms to CIM/V2) into a database using import mapping.

Parameters

The `OvObsImporter` command recognizes the following parameters and options:

`-h |help`

Displays the command parameters and options described in this list along with a short explanation

`-recohelp`

Displays the further options available for configuring the reconciliation of the data to be imported. Reconciling data before importing it can greatly reduce the amount of data, which needs to be processed. The reconciliation process filters out redundant information so that only the changes in your data are imported, thus reducing the load on your resources.

`-parallelhelp`

This option retrieves information on parallel (scalable) processing, where appropriate.

`-v |version`

Displays the version of the command

`<username> <password>`

The name and the password of the Object-server account with which you want to import the data defined in `-data=<XmlFileName>`.

`<server>`

The name of the management server into which you want to import the data defined in `-data=<XmlFileName>`. This can take the form “localhost”, if you execute the command on the machine where the server is running.

`-data=<XmlFileName>`

The name of the XML file containing the data which you want to import. The format of the XML file must conform to CIM/V2 standards.

Note that spaces in a file name or path are not allowed unless the complete path is enclosed in quotes, for example: `"C:\Program Files\HP Openview\filename.xml"`

`[-debug]`

Displays detailed information about the command progress

`[-logfile=<filename>]`

The name of the file to which you want to log information about the import operation

`[-tempdir=<directory>]`

The name and path of the directory in which you want to store temporary information required by the import operation

`[-form]`

Displays command progress in graphical form

`[-mapping=<mapping>]`

The import-mapping name exactly as you entered it in the `Name` field when using the Import-Mapping dialog in the HP OpenView administrator console to set up mappings between internal and external entity types. The import-mapping name is used to identify which external application the import map refers to. It is recommended that you use a similar name for the import-mapping configuration file, and integration account.

`[-recofile=<filename>]`

The name of the previous data-exchange file you are using for comparison.

`[-changefile=<filename>]`

The file name to which you want to write the list of changes found by the command when it reconciles the contents of the current data-exchange file and the data-exchange file specified in `[-recofile=<filename>]`.

`[-sortsize=<Mb>]`

The amount of memory space (in MB) reserved for the processing required to sort the data to be reconciled.

`[-norecosort]`

Do not sort the reconciliation file. This option is only valid if the specified XML files have already been used in a reconciliation process where the `-keepsortedfiles` option was used.

`[-nodatasort]`

Do not sort the data file. This option is only valid if the XML files have already been used in a reconciliation process where the `-keepsortedfiles` option was used.

`[-notimestamp]`

Do not add a timestamp to the data file.

`[-keepsortedfiles]`

Replace the original files with the sorted files. This option is only valid if the XML files have already been used in a reconciliation process where the `-keepsortedfiles` option was used.

`[-norecoimport]`

Do not import the changes found when comparing current and previous data-exchange files. You should use this option only when you want to create a change list.

`[-parallel=<`*`drive`*`>]`

Master and servant systems- the letter associated with the network drive mapped to the shared directory on the master system that is configured to store the information for the parallel-import process.

`[-polltime=<`*`milliseconds`*`>]`

Master and Servant - the elapsed time in milliseconds between two separate polls from servant systems to the shared folder on the master system.

`[-xmlsize=<`*`Kb`*`>]`

Master system only - the size in kilobytes of the separate XML chunks which the master system sends to the servant systems for processing.

`[-servanttimeout=<`*`seconds`*`>]`

Master system only - the maximum time the master waits for a servant to complete the processing of one chunk and send it back.

`[-stopservants]`

Master system only - stops all servants after the parallel-import task has completed.

AUTHOR

`OvObsImporter` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show how to use the `OvObsImporter` command and some of its options to control the import of data into the management-server database.

- To import data from the file `organization.xml` to a server running locally and write progress information to the file "`org-imp-18-03-2005`", where the user/password is "migration/migration" and use mapping defined in ITSM_organization:

  OvObsImporter migration localhost -data=organization.xml -mapping=ITSM_organization -logfile="org-imp-18-03-2005"

- To import data from the file `organization.xml` to a server running locally using a reconciliation file with the time stamp "org_200103011615.xml" and writing information to the file "`org_log.txt`":

  OvObsimporter migration localhost -data=organization.xml -mapping=ITSM_organization -logile=C:\log\org_log.txt -recofile=org_200103011615.xml -norecosort -changefile=C:\log\organization.txt

  Note that the `-norecosort` option can be used because a previous reconciliation process has already sorted this file once.

- Parallel importing on the Master machine:

  OvObsImporter migration localhost -data=organization.xml -mapping=ITSM_organization -parallel=Z: -xmlsize=500

- Parallel importing on the Servant machine:

```
OvObsImporter migration localhost -parallel=Z:
-logfile="servant-run-1" -debug
```

SEE ALSO

OvObsExporter(1m)

COPYRIGHT

OvObsLoadObject(1)

NAME

OvObsLoadObject – Insert, update, or delete objects in the HP OpenView database which relate to events that occur in external systems.

SYNOPSIS

```
OvObsLoadObject  -h |--help

OvObsLoadObject  -v |--version

OvObsLoadObject  -f|--configfile <config file>
-a|--account username/password

OvObsLoadObject -s|--server <hostname[.domainname]>

OvObsLoadObject -p|--port <1024 to 65535 default is
30980>

OvObsLoadObject -x|--mapping <name of mapping>
-c|--class <Classname>

OvObsLoadObject [-l|--logfile <log file name>
-e|--errlogfile <error log file>]

OvObsLoadObject [-o|--onfail
<server:port>,<server:port>... [-o|--onfail
<server:port>,<server:port>...] ]

OvObsLoadObject [-m|--modus <INSERT|DELETE|UPDATE>]

OvObsLoadObject [-r|--resend]

OvObsLoadObject [-v|--keyval <key=value>,<key=value>,...
[-v|--keyval <key=value>,<key=value>,...]]
```

DESCRIPTION

`OvObsLoadObject` allows you to integrate data received from external applications into HP OpenView. The `OvObsLoadObject` command is a command-line interface which allows external applications to insert, update, or delete objects in the HP OpenView database as a result of events that occur in external systems. Applications can use the command

to insert, update, or delete individual objects whenever an event or events occur in the external system, or automate the process so that the database is kept up to date transparently.

If you use the -v or --keyval option to specify a list of key=value pairs, separate the entries in the list with a comma. No spaces are allowed in the list. These rules also apply to the list of backup servers specified with the -o|--onfail option. After processing the configuration file and the command-line, the following values are mandatory: server, port, account, mapping, and class.

Parameters

The `OvObsLoadObject` command recognizes the following parameters and options:

`-h|--help`

Displays the parameters and options described in this list along with a short explanation

`-v|--version`

Displays the version of the command

`-a|--account username/password`

Specify the user name and password of a database account which has the permissions required to manipulate the inserted object.

`-s|--server <hostname[.domainname]>`

Specify the name of the HP OpenView management server to which you want to send the object. Do not specify the name of the database server, if the database is running on a remote server.

`-p|--port <PortNumber>`

Specify the port number of the HTTP listener on the HP OpenView management server. The default port number is 30980; if your system is configured differently and you need to specify a port number manually, choose a number between 1024 and 65535.

`-x|--mapping <name of mapping> [-c|--class <Classname>]`

Import the external data using the specified mapping and object class. Import maps and object classes must already exist and be known to the management server: they are defined using the data-exchange feature in HP OpenView and enable you to ensure that the data from external events are interpreted correctly during the import process.

`[-f|--configfile <config file>]`

Imports the object according to the values specified in the names configuration file and using the

`[-l|--logfile <LogfileName> -e|--errlogfile <ErrorLogFileName>]`

Write progress information about the event-data import process to <LogFileName>; write information about errors that occur during the import process to <ErrorLogFileName>.

`[-o|--onfail <server:port>,<server:port>...]`

Specifies the names of the management servers (and corresponding port numbers) to contact should the connection to the primary management server specified in --server fail for any reason. You can specify multiple lists of servers and each list can contain multiple servers. Spaces are not allowed in a list of servers; the “server:port” pairs in the list are separated by a comma.

`[-m|--modus <INSERT|DELETE|UPDATE>]`

Specify what you want to do with the data from the external event; you can choose between inserting a new object in the database and updating or deleting an existing object.

`[-r|--resend]`

Resend data that failed to reach the management server in a previous operation. The resend option can only be used in conjunction with the `-f` option, where the file specified is the error-log file containing details

of the failed data-import operations. If the resend operation is successful, the SEND= option in the error-log file is changed from “false” to “true”.

`[-v|--keyval <key=value>,<key=value>,...]`

Specify the properties of the object you want to send to the HP OpenView management server; these values can also be defined in a configuration file. The list introduced by the `-v` option contains values that must be separated by a comma and take the form “key=value”, for example: “priority=Low,ci=HPOV”. Spaces are not allowed between the value pairs.

If you choose to read arguments from a configuration file rather than the value list on the command line, note that the configuration file must be formatted in the same way as an MS Windows .INI file. MS Windows .INI files are divided into sections where each section header is a line that contains the section name between square brackets, for example: [OV_EVENT].

AUTHOR

`OvObsLoadObject` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show how to use the `OvObsLoadObject` command and some of its options to control the import of external-event data into the HP OpenView database.

- To insert (`-m`) external event data into the primary management server `server1` (or the backup servers `server2` or `server3`) all listening on port 30980 using the database account/password combination specified in `-a` and formatted using the import mapping and class specified in `-x` and `-c`, logging the insert process to `OvObsLoadObject`.log and using the values specified in the value list `-v`:

  OvObsLoadObject -l OvObsLoadObject.log -s server1 -p 30980 -a system/hpov -x external_event -c incident -m insert -v description="test

```
1234",status=s1,event_id=654321,information="my
info",impact=2,priority=Low,ci=HPOV -o
server2:30980,server3:30980
```

- To import external event data by defining parameters in the configuration file `OvObsLoadObject.conf` and then calling the configuration file from the command line with the `-f` option:

  ```
  OvObsLoadObject -f OvObsLoadObject.conf
  ```

- To resend event data logged in the file `ErrorLogFile` after the send operation failed at the first attempt:

  ```
  OvObsLoadObject -f ErrorLogFile -r
  ```

SEE ALSO

OvObsImporter(1m), OvObsExporter(1m)

COPYRIGHT

A Integration With Network Node Manager

This appendix describes the integration between HP OpenView Service Desk and HP OpenView Network Node Manager (NNM).

Features

NNM provides tools for fault, configuration, and performance management of multi-vendor TCP/IP and IPX/SPX networks. By integrating NNM with Service Desk, you can do the following:

- Automatically forward events from NNM to Service Desk.

  NNM events are stored as incidents in the OpenView database.

- Import nodes discovered by the automatic node discovery mechanism of NNM into Service Desk.

  Discovered NNM nodes are stored as configuration items in the OpenView database.

Creating Incidents in Service Desk

This integration uses the `OvObsLoadObject` object loader utility (see Chapter 2) to automatically forward event information from NNM to Service Desk.

Any event can be configured to be forwarded to Service Desk, including `OV_ARPChgNewPhysAddr` , `OV_Bad_Subnet_Mask`, `OV_NS_PerformWarn`, `OV_NS_PerformErr`, `OV_dataWarehouseMaintError`, `OV_Node_Down`, `OV_Segment_Critical`, and `OV_Network_Critical`.

Importing Nodes Into Service Desk

Node and network information from Network Node Manager can be extracted and imported into Service Desk as configuration items. Data exchange is used to extract the information from Network Node Manager and import it into the OpenView database. The extraction is done using a Java DataBase Connectivity link (JDBC) to the NNM database. The NNM data must first be exported to a data warehouse in an Oracle or SQL Server database (see “Exporting NNM Data to a Data Warehouse” on page 91 for additional information).

NOTE To prevent errors when importing configuration items, the Search Code field of a configuration item must be a non-unique property. See “Make the Search Code Field of Configuration Items Non-Unique” on page 93 for instructions.

Installation

Prerequisites

The NNM integration must be installed on the computer that runs the NNM server. The NNM Integration installs all the software and the utilities that are needed by the integration: `OvObsLoadObject`.

The Service Desk management server must run on a computer different from the NNM server. The HTTP protocol must be enabled on the Service Desk management server.

To enable the HTTP protocol:

1. Log on to the system on which the Service Desk management server is installed.
2. Depending on the operating system, run one of the following commands to launch the Server Configuration program:
   - UNIX: `/opt/OV/bin/OvObsServerSettingsEditor`.
   - Windows: Start>Programs>HP OpenView>Server Settings.
3. Navigate to the Protocols tab.
4. Select HTTP from the **Protocol** drop-down list.
5. Select the **Enable this protocol** check box.
6. From a command prompt, use the `ovc` command to restart the ovobs process:

   ```
   ovc -restart ovobs
   ```

Before You Start the Installation Procedure

Before you Start the Install Procedure:

- Read the contents of Chapter 2 “Preparing to Install HP OpenView Service Desk 5.0” of the HP OpenView Service Desk Installation Guide, including the section “HP OpenView Installer – Overview” – this contains a description of the HP OpenView Installer and associated screens.

- Ensure that you have the appropriate permissions required to install the software. You need administrator (Windows) or root (UNIX) privileges to install this software.

Installation Procedure

1. Log on to the host system on which the NNM management server is installed.
2. Run the executable installation file.

   The file is named `<UnpackDir>/nnmintegration_<revision>_setup.bin` (UNIX) or `<UnpackDir>\nnmintegration_<revision>_setup.exe`, (Windows) where `revision` refers to the build identifier, for example 5.12.345.
3. The **Introduction window** of HP OpenView Installer is displayed. If you previously installed any Service Desk component, the install wizard prompts you to use the install configuration file or decline to use it. For information on this file, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.
4. The HP OpenView Installer starts and guides you through the install process. For a full description of the standard install procedure for Service Desk components, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.

Installed Files

The following files are installed:

- `<DataDir>/conf/sd/ovsdnnmevent.conf`

  Configuration file for the object loader. This file contains information on how to connect to the Service Desk management server. It also describes which import mapping to use for inserting an incident into Service Desk. See Chapter 2 for a detailed description of this file.

- `<DataDir>/conf/sd/ovsdnnmexport_oracle.conf`

  Configuration file for the data exchange exporter. This file contains information on how to connect to the NNM data warehouse running in an Oracle database. It also describes what data will be extracted from the NNM database. See Chapter 1 for a detailed description of this file.

- `<DataDir>/conf/sd/ovsdnnmexport_sqlserver.conf`

  Configuration file for the data exchange exporter. This file contains information how to connect to the NNM data warehouse running in a Microsoft SQL Server database. It also describes what data will be extracted from the NNM database. See Chapter 1 for a detailed description of this file.

- `<DataDir>/conf/sd/ovsdnnmconfigdata.xml`

  Configuration exchange file that must be uploaded into Service Desk. See “Configuration” on page 91. This file contains the Service Desk configuration data for the NNM integration.

- `<InstallDir>/newconfig/conf/sd/ovsdnnmevent.conf`

  This file is a copy of the original `<DataDir>/conf/sd/ovsdnnmevent.conf` file. It can be used to restore the configuration file to the original state.

- `<InstallDir>/newconfig/conf/sd/ovsdnnmexport_oracle.conf`

  This file is a copy of the original `<DataDir>/conf/sd/ovsdnnmexport_oracle.conf` file. It can be used to restore the configuration file to the original state.

- `<InstallDir>/newconfig/conf/sd/ovsdnnmexport_sqlserver.conf`

  This file is a copy of the original `<DataDir>/conf/sd/ovsdnnmexport_sqlserver.conf` file. It can be used to restore the configuration file to the original state.

- `<InstallDir>/newconfig/conf/sd/ovsdnnmconfigdata.xml`

  This file is a copy of the original `<DataDir>/conf/sd/ovsdnnmconfigdata.xml` file. It can be used to restore the configuration file to the original state.

- `<InstallDir>/lbin/sd/ovsdexchnnmnodes.sh` (UNIX only)

  This shell script can be used to exchange nodes between the NNM data warehouse and the CMDB of Service Desk. This script can only be used on the UNIX server that runs both NNM and a Service Desk client. It uses the command line interfaces of the data exchange exporter and importer. You need to specify the database type of the NNM data warehouse in the shell script: Oracle or SQL Server.

Configuration

This section describes the configuration tasks to be performed in Service Desk and on the NNM server.

Exporting NNM Data to a Data Warehouse

Network Node Manager must have a data warehouse installed on an Oracle or SQL Server database to be able to access it with JDBC.

NOTE For detailed information on how to create a data warehouse for NNM, refer to the NNM documentation: *Reporting and Data Analysis with Network Node Manager*. The procedures that follow are to be used as a guideline only.

For NNM on Windows for Oracle or Microsoft SQL Server:

1. Create an Oracle or MS SQL Server account called ovdb with password ovdb.
2. Create an Oracle or MS SQL Server ODBC connection named NNM to the Oracle or MS SQL Server account ovdb.
3. Create the Network Node Manager table structure in the ovdb account, with the following command:

   ```
   C:\> ovdwconfig.ovpl [-rdb ODBC datasource] [-u user]
   [-password password] [-type embedded|msSqlSrvr|oracle]
   [-load]
   ```

   For example, for an Oracle database:

   ```
   C:\> ovdwconfig.ovpl -rdb NNM -u ovdb -password ovdb
   -type oracle -load
   ```

   For example, for a Microsoft SQL Server database:

   ```
   C:\> ovdwconfig.ovpl -rdb NNM -u ovdb -password ovdb
   -type msSqlSrvr -load
   ```

4. The following command loads the topology data into the ovdb account using the ODBC connection:

```
C:\> ovdwtopo -export -rdb nnm
```

For NNM on UNIX for Oracle:

1. Create an Oracle account called ovdb with the password ovdb.
2. NNM installs and configures the appropriate ODBC driver for the Oracle database. The ODBC data source is configured in the `/etc/opt/OV/share/conf/analysis/system_odbc.ini` file. The default ODBC connection is named `OVoracle` and is pointing to the OpenView database using the Oracle Net8 alias ov_net. Change the Net8 alias ov_net to point to the database where you created the Oracle account in step 1.
3. Create the Network Node Manager table structure in the ovdb account, with the following command:

   ```
   # ovdwconfig.ovpl -rdb OVoracle -u ovdb -password ovdb
   -type oracle -load
   ```

4. The following command will load the topology data into the Oracle ovdb account using the ODBC connection:

   ```
   # ovdwtopo -export -rdb OVoracle -v
   ```

Configuration of Service Desk

The process of configuring Service Desk consists of the following procedures:

1. Copy and Upload Configuration Data into Service Desk.
2. Make the Search Code Field of Configuration Items Non-Unique.
3. Modify the Event Configuration File on the NNM Server.
4. Modify and Copy the Data Exchange Exporter Configuration File.
5. Copy the JDBC Driver to the Service Desk Client Computers.

Copy and Upload Configuration Data into Service Desk

Copy the file `<DataDir>/conf/sd/ovsdnnmconfigdata.xml` from the NNM server to a Service Desk client computer.

To upload the configuration data into Service Desk:

1. Start the OpenView console and log on as system administrator.

2. Select **File>Configuration Exchange>Import**.

   The configuration exchange import wizard opens.

3. Click **Next** on the first page of the import wizard, click **Add**, then browse to the directory containing the `ovsdnnmconfigdata.xml` file.

4. Select the `ovsdnnmconfigdata.xml` file and click **Start import**.

5. After importing the configuration data, exit the client.

The uploaded configuration data consists of:

- A NNM integration account for NNM to access Service Desk
- A role that defines the access rights of the NNM account
- An import mapping for creating incidents and configuration items
- A template for creating Incidents
- Templates for creating configuration items (nodes, interfaces, printers, networks, network segments
- Configuration exchange filters
- A configuration exchange filter group

The configuration exchange filter group NNM_config_data contains the filters. You can use this filter group for downloading all NNM configuration data that are stored in Service Desk to a file. This way you can easily transfer your configuration data from one Service Desk deployment (for example, your test environment) to another Service Desk deployment (for example, your production environment).

Make the Search Code Field of Configuration Items Non-Unique

1. In the OV Configuration workspace group, select **Data>Unique Fields**.

2. Double click **Configuration Item**.

   The Unique Fields - Configuration Item window opens.

3. Clear the **Search Code** field.

4. Save the settings and close the window.

Modify the Event Configuration File on the NNM Server

Modify the connection details in the `<DataDir>/conf/sd/ovsdnnmevent.conf file`. In the SERVER entry, replace `mysdserver.mydomain.com` with the fully qualified domain name (FQDN) of the Service Desk management server.

Modify and Copy the Data Exchange Exporter Configuration File

The configuration file you need to modify is dependent upon your choice of the relational database. If you run the NNM data warehouse on an Oracle database, modify the connection details in the `<DataDir>/conf/sd/ovsdnnmexport_oracle.conf` file. If you run the NNM data warehouse on a Microsoft SQL Server database, modify the connection details in the `<DataDir>/conf/sd/ovsdnnmexport_sqlserver.conf` file.

Replace in the URL entry `myNnmDbServer.myDomain.com` with the FQDN of the database server that runs the NNM data warehouse. Replace `myNnmDataWarehouse` with the name of the database instance. This information was specified when you created the NNM data warehouse.

Copy the configuration files `<DataDir>/conf/sd/ovsdnnmexport_oracle.conf` or `<DataDir>/conf/sd/ovsdnnmexport_sqlserver.conf` from the NNM server to each Service Desk client computer that will perform the import of the NNM nodes.

Copy the JDBC Driver to the Service Desk Client Computers

Each Service Desk client computer that needs to be able to import the NNM nodes into Service Desk require the JDBC driver to connect to the NNM data warehouse. Which driver is needed depends on your choice of relational database. You need to copy the JDBC driver from a Service Desk management server to the Service Desk client computer.

If you run the NNM data warehouse on an Oracle database, copy the Oracle JDBC driver `<InstallDir>/nonOV/obs/ojdbc14.jar` from the Service Desk management server to the directory `<InstallDir>/java/drivers` on the Service Desk Client computers.

If you run the NNM data warehouse on a Microsoft SQL Server database, copy the SQL Server JDBC driver `<InstallDir>/nonOV/obs/Opta.jar` from the Service Desk management server to the directory `<InstallDir>/java/drivers` on the Service Desk client computers.

If the Service Desk client runs on UNIX, make sure the jar file has the correct ownership and permissions. The file must be owned by bin and must be in the bin group. Issue the following command from a command line to set the correct ownership:

```
# chown bin:bin <driver>.jar
```

The permissions of the jar file must be read-only for everyone. Issue the following command from a command line to set the correct permissions:

```
# chmod 444 <driver>.jar
```

Configuration of Network Node Manager

The following configuration tasks need to be performed in Network Node Manager.

1. Add a trusted command.
2. Configure automatic action for SNMP traps.

   You need to configure this automatic action for every SNMP trap that you want to be forwarded to Service Desk.

Add a Trusted Command

By default security measures, NNM only performs automatic actions on trusted commands that are executed with root or administrator permissions. See the `ovactiond` NNM manual reference page for the format and rules to configure NNM to trust an automatic action. Make the `OvObsLoadObject` object loader utility a trusted command for NNM. On UNIX, add the following line to the file `$OV_CONF/trustedCmds.conf/NNM`:

```
sendtoSD=/opt/OV/lbin/obs/loadobject/OvObsLoadObject -f
/var/opt/OV/conf/sd/ovsdnnmevent.conf
```

Make `ovactiond` re-read the files in the trusted commands configuration directory, by running the following command:

```
# xnmevents -event
```

On Windows, add the following line to the file `%OV_CONF%\trustedCmds.conf\NNM`:

```
sendtoSD=C:\Program Files\HP
OpenView\lbin\obs\loadobject\OvObsLoadObject.bat -f
C:\Program_Files\HP_OpenView \data\conf\sd\ovsdnnmevent.conf
```

Make `ovactiond` re-read the files in the trusted commands configuration directory, by running the following command:

```
C:\> xnmevents.exe -event
```

Configure Automatic Action for SNMP Traps

The configuration of SNMP traps can be performed in two ways. You can edit the trapd.conf file with a text editor or you can use the NNM GUI. This section only describes the use of the GUI. For detailed instructions you are referred to the NNM online help and NNM documentation.

1. Start the NNM GUI.
2. Open the Event Configuration window and select an event, or open the Event Browser, select an event and choose the Configure Event menu option.
3. Modify the selected event. Enter the trusted command `sendtoSD` in the **Command for Automatic Action** field. The parameters you pass to the command depend on the event. You can pass the following parameters to the command:

   - event_id (mandatory). This parameter is used to uniquely identify the incident that will be created in Service Desk. The NNM variable $U is very suitable for this purpose. The event_id will be stored in the Source ID field of the incident.

     If you pass a value with event_id that is already used, no new incident is created in Service Desk. Instead, the existing incident in Service Desk is updated.

   - description (mandatory). This is a free-format text string parameter with a maximum length of 255 characters. The message in the **Event Log Message** field is very suitable for this parameter. The message may contain NNM variables. Refer to the NNM online help or NNM documentation for a detailed description of the NNM variables and their values.

NOTE Make sure you surround the message string with double quotes.

- ci (optional). This parameter specifies the node from which the event is received. A relation between the Service Desk incident and the configuration item in the CMDB of Service Desk is created. Many events contain the FQDN of the node in the $2 variable.
- severity (optional). The severity of the event (Critical, Major, Minor, Warning, Normal) can be passed with this parameter. The $s variable contains the severity.
- impact (optional). You can use the same values of the severity to pass to the impact parameter. The values are mapped to the Service Desk values of impact (Top, High, Medium, Low, None). Service Desk will automatically calculate the priority of the Incident based on the value of the impact.
- status (optional). When you omit this parameter the default value Registered is filled in the State of the incident. You can overrule the default value by passing a different value to this parameter. Possible values are Registered, In Progress, Closed or Completed.
- information (optional). This is a free-format text string parameter with a maximum length of 4000 characters. You can pass more information to the incident with this parameter.
- solution (optional). This is a free-format text string parameter with a maximum length of 255 characters. If a solution is known for the received event you can supply the solution with this parameter.

  The syntax of the parameters is in the format of key/value pairs.

  ```
  sendtoSD -v <key>=<value> [-v <key>=<value> …]
  ```

  No spaces are allowed around the equal sign.

4. Optionally you can modify or add sendtoSD parameters. You can map the parameters of sendtoSD to other fields of an incident or add additional parameters that map to selected fields of an incident. When you change the parameters of sendtoSD you must change the import mapping of the events accordingly. Use the Service Desk

client to change the import mapping of NNM events. The nnm import mapping contains the nnm_event external entity that defines the attribute and value mappings. For instructions on how to create or modify the import mappings, see Chapter 2.

Managing Event Storms on UNIX

You can use the object loader queuing tools to handle event storms. Refer to the section “Manage Event Queues” on page 65 for instructions.

Examples

These examples illustrate how to configure NNM events to be forwarded to Service Desk. For each NNM event listed, a command for automatic action is suggested.

- OV_ARPChgNewPhysAddr

  ```
  sendtoSD -v event_id=$U -v description="$6 reports a
  different physical address for $2 than reported by $8,
  changing from $12 to $13" -v ci=$2 -v severity=$s -v
  impact=$s
  ```

- OV_Bad_Subnet_Mask

  ```
  sendtoSD -v event_id=$U -v description="Inconsistent
  subnet mask $9 on interface $7, should be $10" -v ci=$2
  -v severity=$s -v impact=$s
  ```

- OV_NS_PerformWarn

  ```
  sendtoSD -v event_id=$U -v description="Your Name
  Services is performing poorly, possibly misconfigured.
  $3 of $4 took $5 seconds to resolve." -v severity=$s -v
  impact=$s
  ```

- OV_NS_PerformErr

  ```
  sentoSD -v event_id=$U -v description="Your Name Services
  are performing poorly, likely misconfigured.  The average
  for the past $3 lookups took $4 milliseconds, expect less
  than $5 milliseconds." -v severity=$s -v impact=$s
  ```

- OV_dataWarehouseMaintError

  ```
  sendtoSD -v event_id=$U -v description="Data warehouse
  maintenance program ($4) exited with a non-zero return
  code of $5 and the following message: $6" -v ci=$2 -v
  severity=$s -v impact=$s
  ```

- OV_Node_Down

  ```
  sendtoSD -v event_id=$U -v description=" Node Down
  Capabilities: $8  Root Cause: $9 $10" -v ci=$2 -v
  severity=$s -v impact=$s
  ```

- OV_Segment_Critical

```
sendtoSD -v event_id=$U -v description=" Segment
critical" -v ci=$2 -v severity=$s -v impact=$s
```

- OV_Network_Critical

```
sendtoSD -v event_id=$U -v description=" Network
critical" -v ci=$2 -v severity=$s -v impact=$s
```

User Tasks

You can import NNM nodes into Service Desk in the following ways:

- Use the OpenView console.

  You can only use the OpenView console on a Service Desk client where you copied the data exchange exporter configuration file during the configuration of Service Desk.

  When you use the OpenView console, you create a data exchange task. See “Create a Data Exchange Task” on page 42. Specify the data exchange exporter configuration file that you copied as instructed earlier.

- Use the command line interfaces of data exchange that come with the Service Desk client.

  If the Service Desk client runs on a UNIX NNM server, a shell script `ovsdexchnnmnodes.sh` is provided to assist you in running the command line interfaces. The script is located in the `/opt/OV/lbin/sd` directory on the NNM server.

  Before you run the `ovsdexchnnmnodes.sh` script, change the DATABASE variable in the script to the type of database that runs the NNM data warehouse. Possible values are oracle or sqlserver. You can run the script without any parameters. The logfiles are located in the directory `/var/opt/OV/log`.

You need to repeat this user task periodically to update the Service Desk CMDB with new nodes. Depending on how dynamic your network environment is, you may need to repeat this task weekly or even daily.

B Integration with Internet Services

The details of the integration are explained in following sections.

Integration Possibilities

HP OpenView Internet Services provides tools for fault, configuration, and performance management. By integrating Internet Services with Service Desk, you can do the following:

- Synchronize service hierarchy information between Service Desk and Internet Services
- Enable Service Desk personnel to launch the Internet Services dashboard from the OpenView console to view current service status information in support of incident and problem management processes.
- Enable Service Desk personnel to access Internet Services reports from the OpenView console.
- Install and configure the OVIS metric adapter to collect metric data values for the configured IS metrics. The collected metric data values are automatically delivered to the Service Desk management server and used in availability and compliance calculations.

  For more information about the OVIS metric adapter, see *HP OpenView Service Desk Service Level Manager Guide*.

Configuration

If you use Internet Services to monitor the resources in your IT infrastructure, you can specify IS configuration metrics in the service level agreements you create. IS configuration metrics specify the key performance indicators that Internet Services should be configured to measure for the service level agreement. You can export this information from Service Desk and import it into the Internet Services database.

The service level agreement must be set up with general contract information and must be related to at least one service and service receiver organization.

The IS Configuration Form

For instructions on creating an IS Configuration, see the Service Desk Online Help.

Table B-1 lists details of mandatory and non-mandatory information required in the creation of an OVIS configuration.

Table B-1 OVIS Metric Requirements

Field	Possible values	Default	Mandatory
Description	Unique free text		Yes
Probe type	DNS, HTTP FTP, HTTPS, SMTP, ICMP, NNTP, POP3 RADIUS, WAP	HTTP	Yes
Availability condition	> < = >= <= !=	>	Yes

Table B-1 OVIS Metric Requirements (Continued)

Field	Possible values	Default	Mandatory
Availability objective	Rational number between 0% to 100%	100%	Yes
Response time condition	> < = >= <= !=	<	Yes
Response time objective	Time in seconds, a natural number greater or equal to 0	5	Yes
Service	Any service related to the service level agreement that has one or more receivers in common with the receivers registered in the service level agreement.		Yes

Target and Location Information

Table B-2 lists details of location information that can be supplied with an OVIS configuration. All location information is non-mandatory.

Table B-2 OVIS Configuration Location Information

Field	Possible values	Default
Name	Any string value	
Interval	Any natural number	300 secs

Table B-2 OVIS Configuration Location Information (Continued)

Field	Possible values	Default
TImeout	Any natural number	20 secs

Table B-3 lists details of non-mandatory target information that can be supplied with an OVIS configuration. All target information is non-mandatory.

Table B-3 OVIS Metric Target information

Field	Possible values	Default
Host	Any string value	
Port	Any natural number	0
URL	Any string value	

IS Configuration File

For instructions on exporting an IS configuration, see the Service Desk Online Help.

Example B-1 displays a sample IS configuration file.

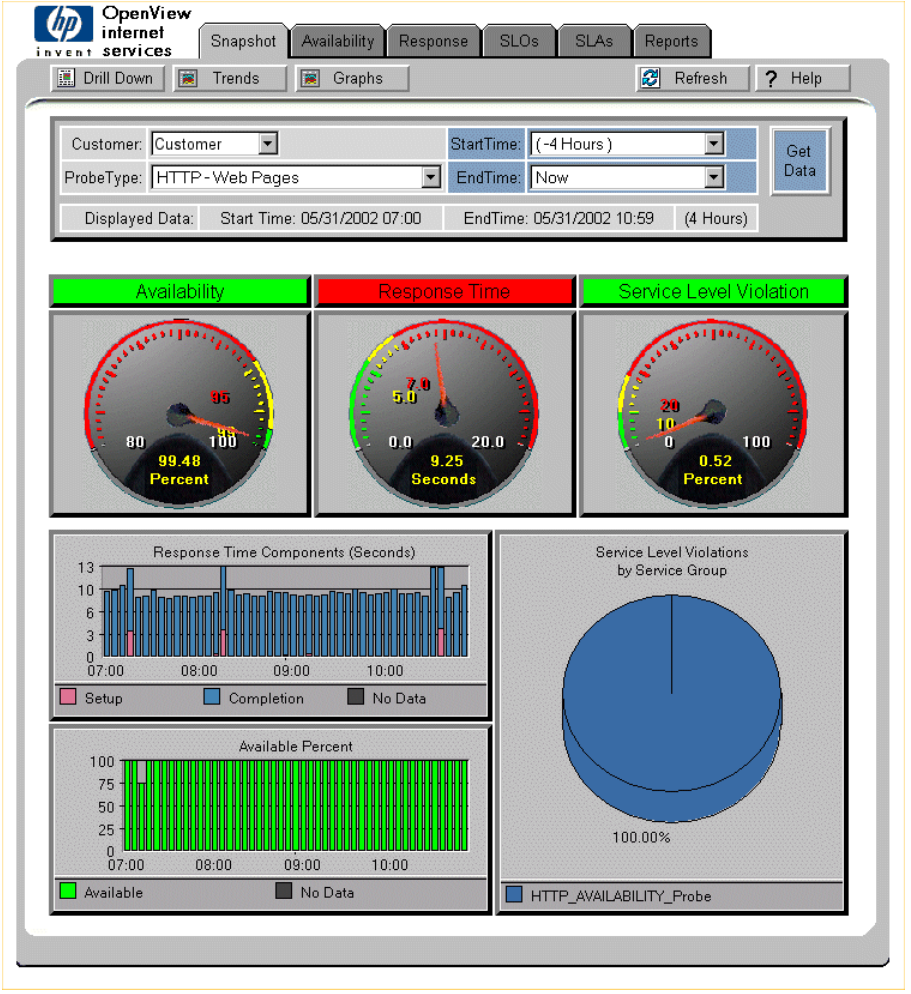
Example B-1 Sample IS Configuration File

```
<CUSTOMERLIST>
 <CUSTOMER name="Invention Incorporated">
  <SERVICE id="SAP_HTTP" probe="HTTP">
   <OBJECTIVE metric="AVAILABILITY" condition=">"    servicelevel="100" />
   <OBJECTIVE metric="RESPONSE_TIME" condition="<"    servicelevel="5" />
   <TARGET host="London" port="0"
   urlfile="http://<management_server>.hpov.com" />
   <LOCATION id="London" interval="300" timeout="20" />
  </SERVICE>
 </CUSTOMER>
</CUSTOMERLIST>
```

Internet Services Dashboard Launch Options

The URLs used to specify context for the Internet Services dashboard are formatted as form-data (or query-data). This is a standard format for retrieving information from a web page to a server-side program. Figure B-1 on page 108 shows an example of a dashboard.

Figure B-1 Example of an Internet Services Dashboard



You can configure a smart action to launch the Internet Services dashboard from the OpenView console. For instructions, see the Service Desk Online Help.

Table B-4 lists the parameters that can be supplied in the smart action.

Table B-4 Name - Value Parameters for Dashboard Launch

Name	Description	Possible Values
Content	Specifies which tab of the dashboard is displayed	Snapshot, Response, Availability, Details, Trend, Reports, Login
Metric	Specified in the Drill Down Details screens	You can select "All Metrics" or a specific metric from the drop-down Metric box. For each measured target, details are presented on each metric selected that is provided by the probe. The metrics will vary based on the type of probe.
Level	You can increase or decrease the level of detail in the reports using the Level drop-down box.	1 - 8. See Table B-5, “Levels of Reporting,” on page 112
Customer	Specified in all screens.	You can select "All Customers" or a specific customer from the drop-down Customer box.

Table B-4 Name - Value Parameters for Dashboard Launch (Continued)

Name	Description	Possible Values
StartTime	Used to filter data for different amounts of time summarized into the display. As you change from one tab to another, your selected data display is preserved. Each display shows the time frame of data actually included in that display. This time frame may be less than was requested depending on how much data has actually been collected.	The time periods may be expressed as relative or explicit. Negative integers are used to denote relative time. For example a value of -4 would mean the last four hours. When relative time is used endTime must have a value of -1. An explicit time period is specified as a positive integer. The values are based on coordinated universal time (UTC) (Number of seconds elapsed since midnight (00:00:00), January 1, 1970).
EndTime	see starttime	The endTime value should always be larger then the startTime value
ProbeType	Used to filter data for different services. You can select a probe type to display the data only for that type of service being probed. As you change from one tab to another, your selected data display is preserved.	You can select "All Services" or a specific service from the drop-down ProbeType box. The values should be specified as they appear in this drop-down list. It should also be noted that a single space appears before and after the hyphen in each of the list items (i.e. "HTTP - Web Pages")

Table B-4 Name - Value Parameters for Dashboard Launch (Continued)

Name	Description	Possible Values
DisplayType		Used to select whether you want to see the data as averages or as averages and time series display.Use this feature by clicking the down-arrow at the right of the box and select from the list provided. As you change from one tab to another, your selected data display is preserved.
Pswd	Used with restricted views to specify the password of a login session. This can only be used when 'content' is set to 'login' and a valid 'customer' value has been specified.	
-customers	Legacy format (OVIS 3.0 and later) to specify the organization for which to show the measurement results.	
-c	Legacy format (OVIS 3.0 and later) to specify the tab page to be shown.	
-i	Legacy format (OVIS 3.0 and later) to specify the IS service to be shown.	

The Level parameter listed in Table B-4 specifies the level of reporting for the Internet Services dashboard. Table B-5 explains the reporting levels available:

Table B-5 Levels of Reporting

Detail Level	Time-Series Graph	Table Summary
1	Customer	
2	Probe Types	Service group
3	Service groups	
4	Host summaries	Host system
5	Target summaries	Target
6	Target details	
7	Probe location	Probe location
8	Location details	

OVIS Integration Example

In this example of using the Internet Services integration, a service level manager configures a service level agreement covering the provision of a web service. IS configuration details suitable for monitoring the performance characteristics of the web service are added. The service level manager exports the IS configuration details to an XML file. The details are then imported into the Internet Services database. A Service Desk specialist accesses the Internet Services dashboard to view web service performance information collected by the configured Internet Services probes.

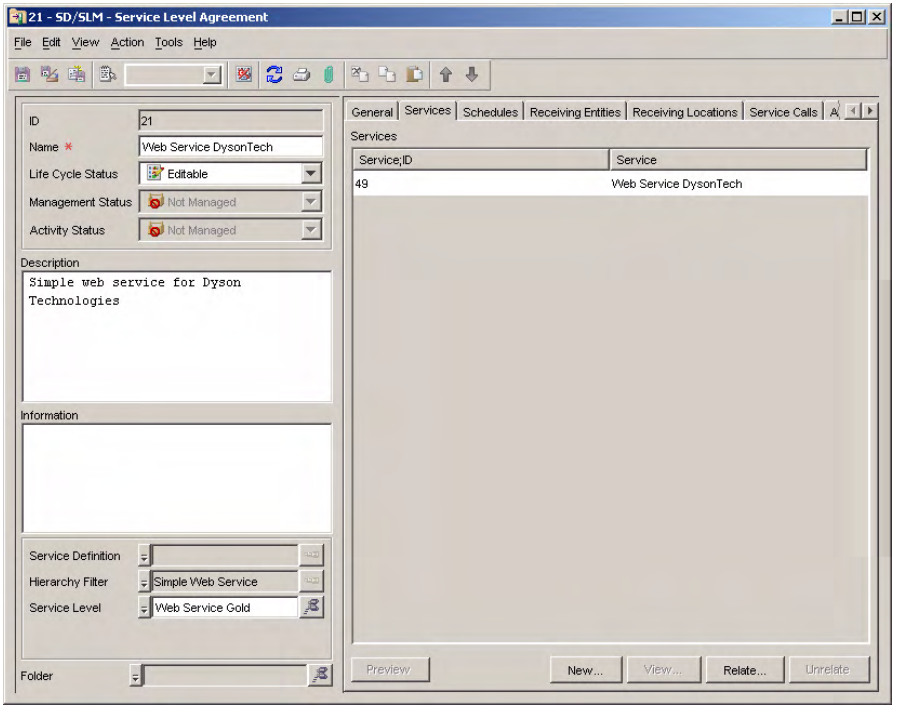
Configure a Service Level Agreement

The service level manager creates a new service level agreement, enters basic details such as the name and description, the actual start and actual finish dates, and the evaluation period. For information on describing service level agreements, see the Service Desk Online Help.

Because the Service level agreement is to be placed under SLM management, the service must be based on a service definition or a hierarchy filter. In this example, the service is based on a hierarchy filter, which must also be selected in the service level agreement.

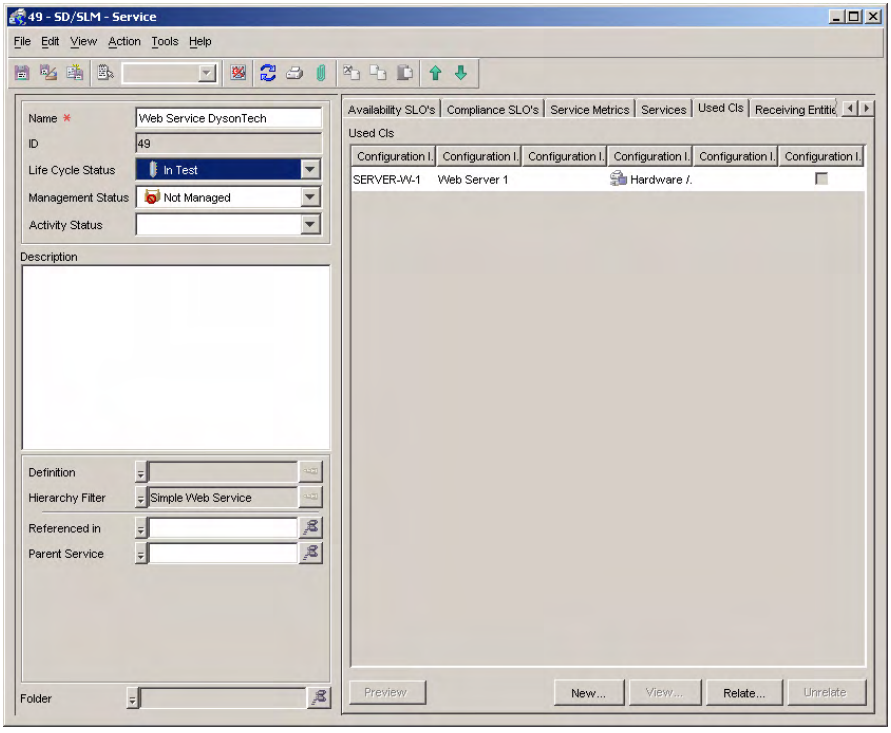
The service level manager also relates the web service to the service level agreement:

Figure B-2 Service Related to the Service Level Agreement



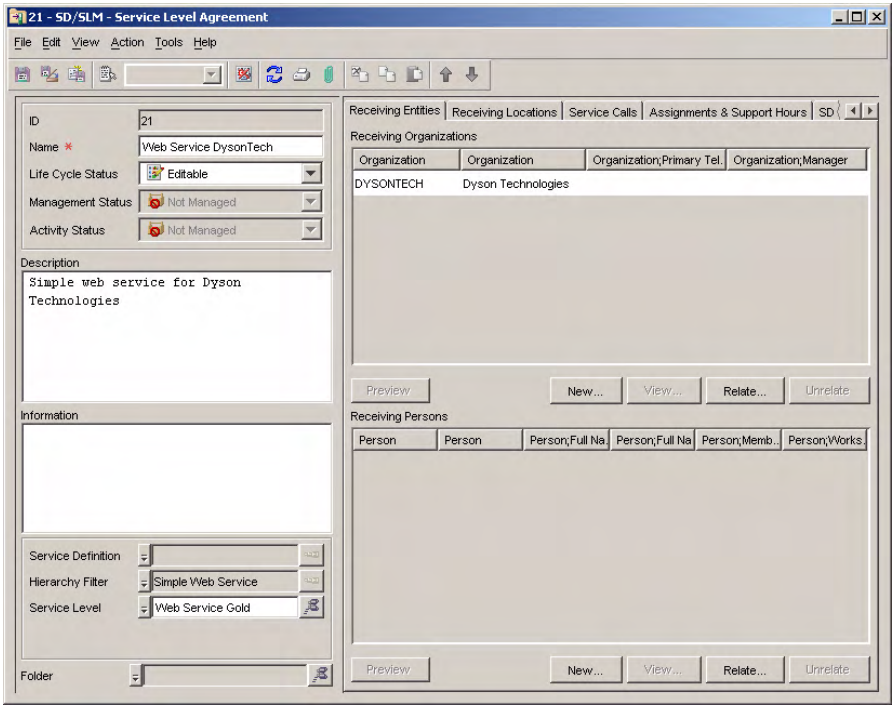
In this example, the web service uses one configuration item representing the web server:

Figure B-3 Used Configuration Item



In this example, the web service is to be supplied to one receiving organization. You should relate the receiving organization to the service and the service level agreement. For further information about supplying services to customers, see the Service Desk Online Help.

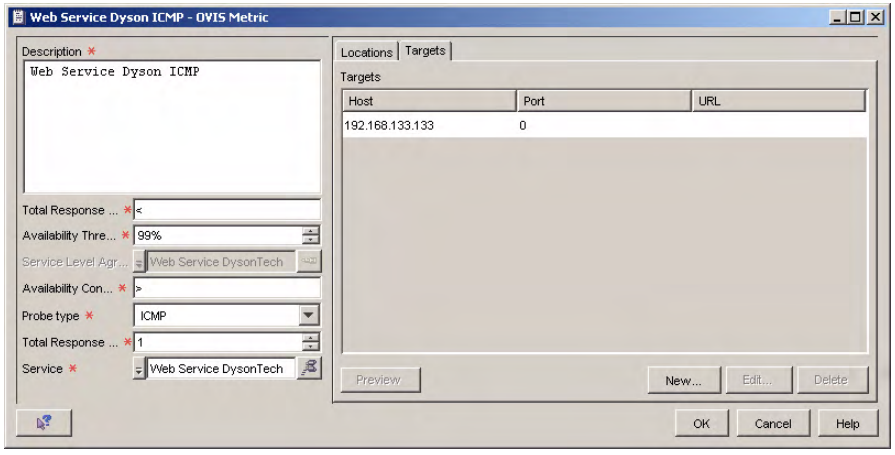
Figure B-4 Receiving Organization



Define IS Configurations

The service level manager decides that Internet Services should be configured to monitor the availability of the web server, and creates an IS configuration that uses the Internet Services ICMP probe. The web server is specified as the target for the probe.

Figure B-5 **ICMP Probe**



The service level manager decides that Internet Services should be configured to monitor the response time for serving up a particular URL on the web server, and creates an IS configuration that uses the Internet Services HTTP probe. The URL is specified as the target for the probe.

Figure B-6 HTTP Probe

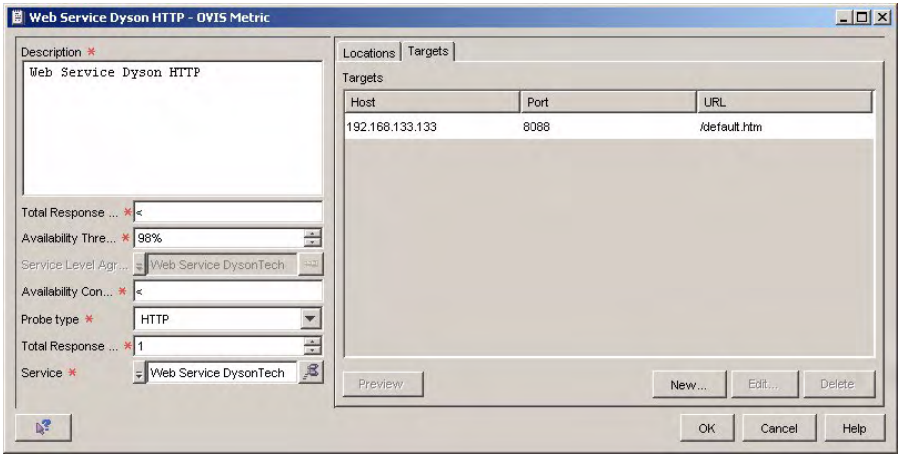
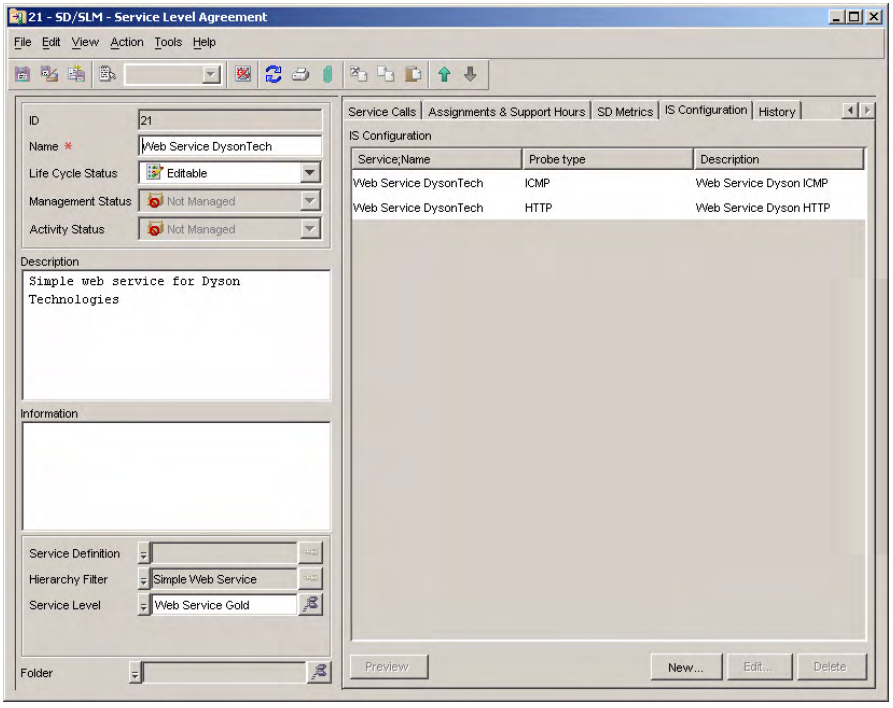


Figure B-7 shows the list of IS configurations that have been added to the service level agreement.

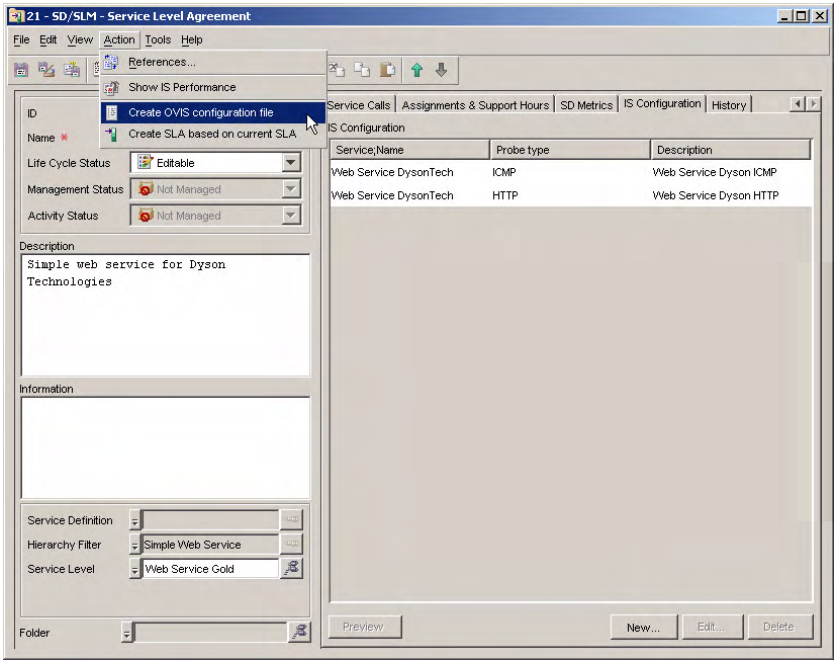
Figure B-7 List of IS Configurations



Create an OVIS Configuration File

The service level manager now issues the command to export the IS configuration information to an XML file.

Figure B-8 Creating the OVIS Configuration File



Service Desk displays a message confirming the location of the created file.

Figure B-9 OVIS Configuration File Confirmation Message

Load the Configuration into Internet Services

The service level manager now imports the OVIS configuration file into Internet Services. This is done by first stopping the HP Internet Services, IIS Admin, and Reporter services using the net stop command, then using the iopsload command to load the configuration file into the Internet Services database, then restarting the stopped services using the net start command.

Example B-2 shows the commands used to import the IS configuration file created in the previous step of this example. For further information on how to load configuration files into Internet Services, refer to the Internet Services documentation.

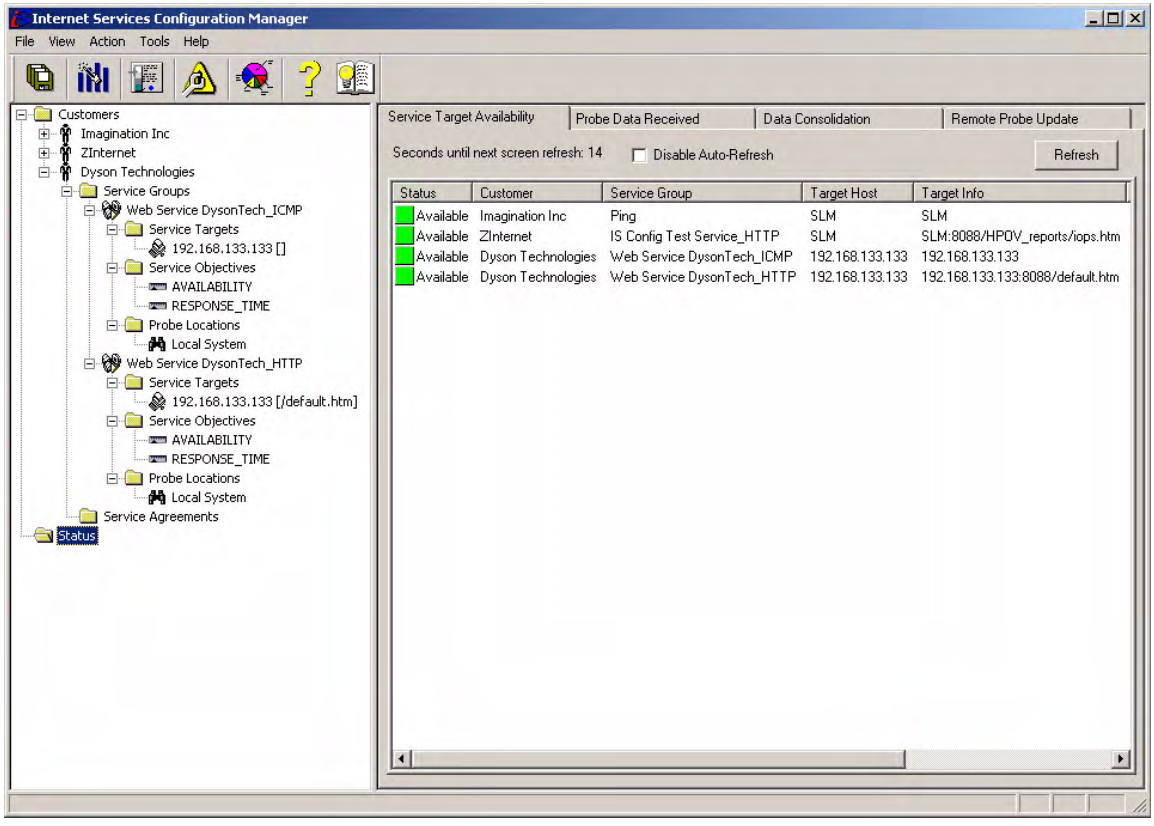
Example B-2 Loading the IS COnfiguration into Internet Services

```
C:\ net stop "HP Internet Services"
C:\ net stop "IIS Admin" /y
C:\ net stop "Reporter"
C:\ iopsload -load "Web Service DysonTech_ID21.xml"

Begin loading configuration file Web Service DysonTech_ID21
Syntax is good
Processing configuration file "Web Service DysonTech_ID21.x
Added    1 customers
Added    2 service groups
Added    2 service targets
Added    4 objectives
Added    2 probe locations

Completed loading configuration file
ExportIOps successful
HP Internet Services service started
Configuration saved to config.dat files and IOps restarted

C:\ net start "Reporter"
C:\ net start "World Wide Web Publishing”
```

The imported configuration is now visible in the Internet Services Configuration Manager. Notice that the Internet Services customers correspond to the names of the receiving organizations in Service Desk,

and each Internet Services service group is a concatenation of the name of the service in Service Desk and the name of the Internet Services probe.

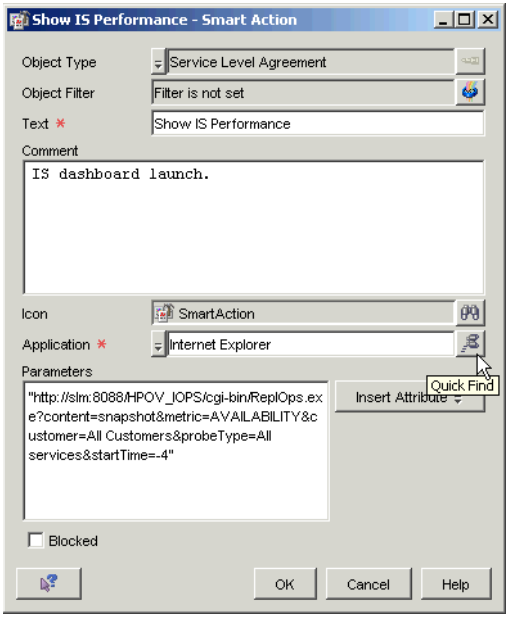
Figure B-10 Configuration Imported into Internet Services



Configure a Smart Action

1. Create a new smart action for service level agreements.

Figure B-11 New Smart Action



2. Type a suitable name for the smart action, such as "Show IS Performance".
3. In the **Application** field, select the application you want this smart action to open. In this case it is 'Internet Explorer'.
4. In the **Parameters** field, enter the URL to be accessed, including the required field values. Surround the URL in double-quotation marks.
5. Save the smart action.
6. Authorize the roles of Service Desk personnel who need to use the smart action.

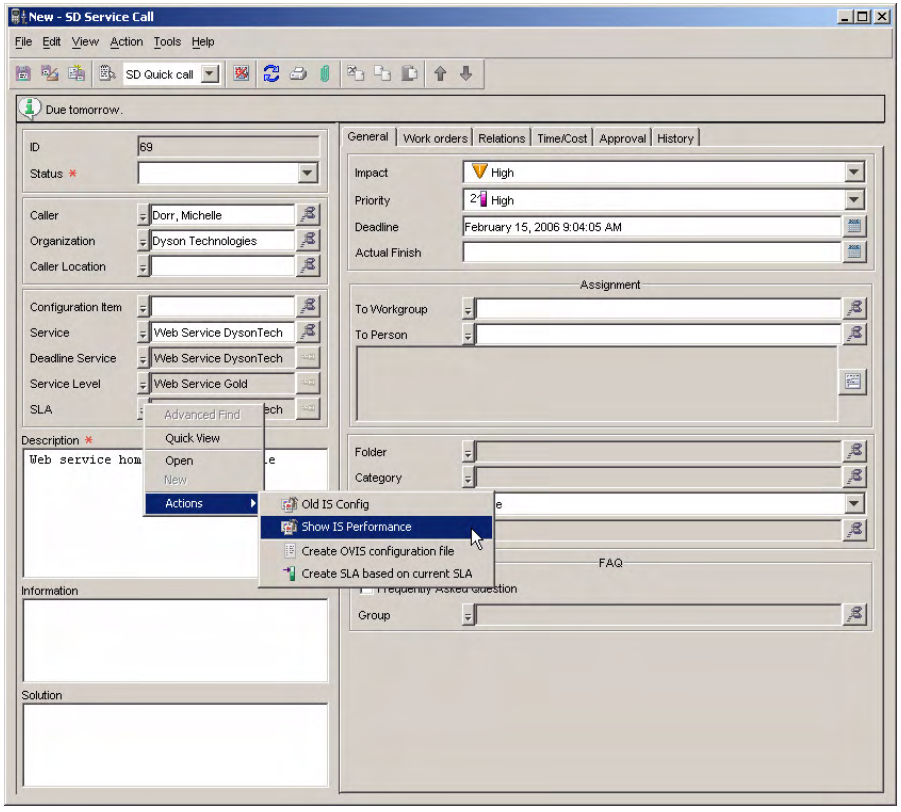
Launch the Internet Services Dashboard from a Service Call

By invoking the configured "Show IS Performance" smart action (see “Configure a Smart Action” on page 123) from a service call, Service Desk users can launch the Internet Services dashboard to display the performance information that has been gathered by Internet Services probes.

1. Open the service call in a form.
2. Click the shortcut menu button to the left of the SLA field, then click **Action → Show IS Performance**.

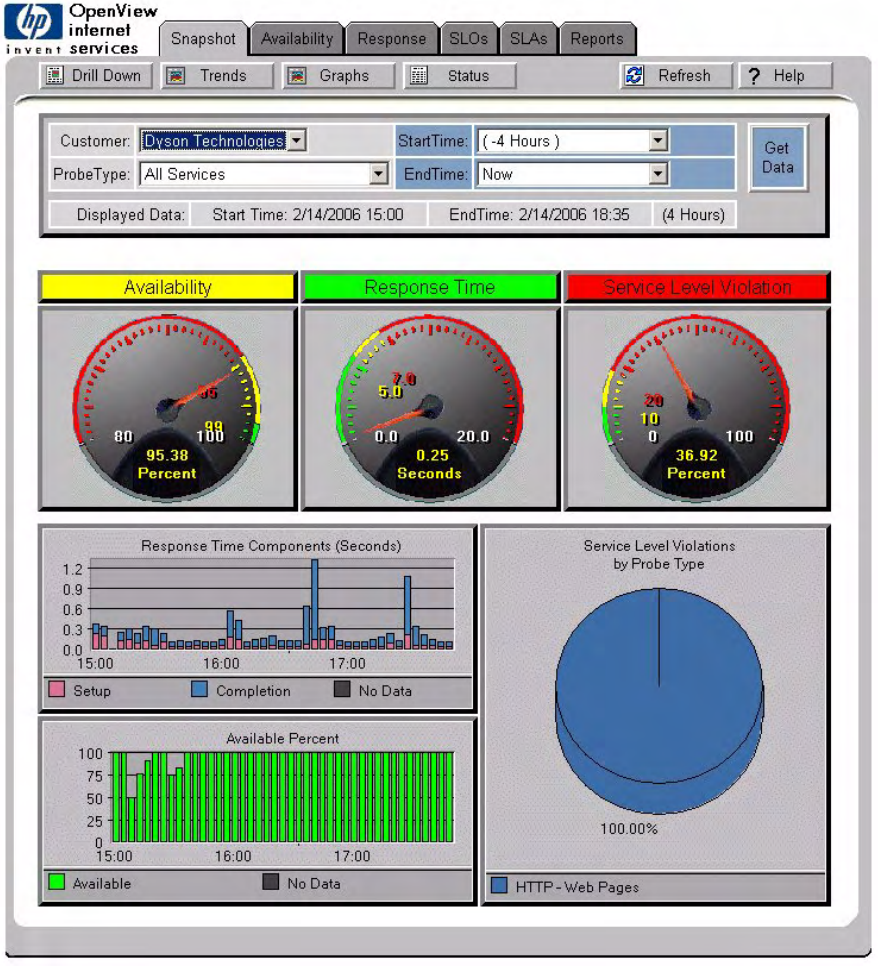
   The command on the Action menu depends on the name given to the smart action (see “Configure a Smart Action” on page 123).

Figure B-12 Invoking the Smart Action



The Internet Services dashboard opens.

Figure B-13 The Internet Services Dashboard



C Microsoft Operations Manager (MOM) Integration

This appendix describes the Service Desk to Microsoft Operations Manager (MOM) integration.

The integration offers the user the following capabilities:

- Exported nodes within MOM are imported as configuration items in Service Desk
- Alerts detected by MOM are transmitted as incidents to Service Desk, and can be mapped against service level agreements
- Changes to incidents in Service Desk are synchronized with the corresponding alert in MOM

Installation

This section describes how to install the MOM integration.

Before you start:

1. Ensure you are familiar with the contents of the Service Desk Installation Guide, Chapter 2 “Preparing to Install HP OpenView Service Desk 5.1”, including the section “The HP OpenView Installer” on page 51 – this contains a description of the HP OpenView Installer and its associated screens.
2. See “System Specifications and Requirements” on page 35 of the Service Desk Installation Guide for general information on install prerequisites for HP OpenView Service Desk components.

To install the MOM integration:

1. Log on to the MOM server using an account with system administrator rights.
2. Run the executable installation file.

   The installation file is located in the Service Desk media root directory. The file is named `momintegration_<revision>_setup.exe`, where `revision` refers to the build identifier, for example 5.00.722.
3. The **Introduction window** of HP OpenView Installer is displayed. If you previously installed any Service Desk component, the install wizard prompts you to use the install configuration file or decline to use it. For information on this file, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.
4. The HP OpenView Installer starts and guides you through the install process. For a full description of the standard install procedure for Service Desk components, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.

The MOM integration installation adds a number of files on your system. The most important files added are the following:

- `%OvDataDir%\conf\obs\loadobject\OvObsLoadObject.conf`

  Additional files associated with the object loader are also installed.

- `%OvInstallDir%\lbin\sd\mom\OvSdMomGetComputers.pl`

  Run this perl script to generate a list of the nodes managed by MOM. This script serves as a basis for exporting nodes to Service Desk.

- `%OvDataDir%\conf\sd\mom\OvSdMomConfig.akm`
- `%OvDataDir%\datafiles\data_exchange\xml\OvSdMomConfig.xml`
- `%OvDataDir%\conf\sd\mom\OvSdToMom.conf`

The Service Desk Agent is also installed. For instructions on starting the Service Desk agent, see Chapter 8, Installing a Service Desk Agent, in *HP OpenView Service Desk Installation Guide*.

Configuration

This section describes the following configuration tasks:

- Upload the Configuration Exchange File
- Modify the Integration Account
- Configure the Object Loader
- Install Forwarding Rule on MOM Server
- Set up Configuration Items in Service Desk to Represent Computers in MOM

Upload the Configuration Exchange File

The configuration exchange file contains configuration settings required by the integration. This section describes how to upload the file.

To upload the configuration exchange file:

1. Do one of the following:
   - Install a Service Desk client on your MOM consolidation server and log on to the Service Desk client as an administrator.
   - Copy the file `%OvInstallDir%\data\datafiles\data_exchange\xml\OvSdMomConfig.xml` to a computer that has the Service Desk client installed and then log on to that client as an administrator.
2. In the OpenView Console, run the command **File>Configuration Exchange> Import** to start the Configuration Exchange import wizard.
3. Import the file `%OvInstallDir%\data\datafiles\data_exchange\xml\OvSdMomConfig.xml`.

Table C-1 lists the configuration settings that the configuration exchange file adds to the database connected to your Service Desk management server.

Table C-1 Imported MOM Integration Configuration Settings

Configuration Setting	Name
Database rules for incidents	Update alert Close incident
Data exchange import mapping	Mom
Integration Account	MOM_server1
Role	mom integration
Incident template	Sd Incident template

After importing the configuration exchange file, you should review the import log file to ensure that the import operation was completed successfully.

Modify the Integration Account

After importing the configuration exchange file, you need to modify the imported MOM integration account. For general information on how to modify accounts, see the Service Desk online help.

Change the account details as follows:

- In the **Account Name** field, change the name from `MOM_server1` to `MOM_`**`hostname`**, where **`hostname`** is the computer name of the system on which the MOM consolidation server is installed.
- In the **Host** field, change the text from `localhost` to the fully qualified hostname of your MOM consolidation server.

You are also strongly advised to change the password of the account from its default value of `Password4OpenV!ew`.

Configure the Object Loader

On the MOM consolidation server, open the object loader configuration file: `%OvDataDir%\conf\obs\loadobject\OvObsLoadObject.conf` in a text editor.

Locate the following lines:

```
ACCOUNT=MOM_server1/MOM_server1
SERVER=localhost
MAPPING=Mom
```

Change the value of the ACCOUNT parameter to the application account and password you have defined for your MOM application account, separated by a forward slash. Change the value for the SERVER parameter to the fully qualified DNS entry of your Service Desk application server.

Install Forwarding Rule on MOM Server

Log on to the MOM Consolidation server and follow these instructions to install the forwarding rule.

To install the forwarding rule on the MOM server:

1. Log on to the Microsoft Operations Manager console.
2. Navigate to **Rules**, **Processing Rule Groups**.
3. Right-click **Processing Rule Groups** and select **Import Management Pack**.
4. Click the **Browse** button, navigate to the directory in which you installed the integration, and open the file `%OvInstallDir%\data\conf\sd\mom\OvSdMomConfig.akm`.
5. For all other options, accept the defaults and click the **Next** button to continue.
6. Click **Finish** on the page after that.

Confirm that a new processing rule group called HP OpenView Service Desk has been imported. This rule group contains a subgroup called Create Incident in Service Desk. The subgroup contains the alert processing rule called Create Incident in Service Desk and the event processing rule called Create Incident in Service Desk. The alert rule forwards events as incidents to Service Desk, but the alert is suppressed

by MOM by default and is packed in its parent alert. The consequence is that no annotation can be written back to this alert by Service Desk. The event processing rule forwards events to Service Desk and enables annotations to be sent from Service Desk to MOM. The alert raised by the event does not get suppressed and packed when the 'suppress duplicate alerts' setting on the Alert suppression tab of the Event Alert Rule is set to false.

You can filter which events are forwarded by altering the processing rules.

After completing all the steps up to this point, you should be able to see alerts (as far as they match the alert criteria and schedule) in MOM represented as incidents in Service Desk. When you open an alert in MOM that corresponds to an incident in Service Desk, you should be able to see the Service Desk ID and the other incident fields registered in Service Desk in the comments section of the alert properties. When you close one of these incidents in Service Desk, the corresponding alert status should automatically change to `Resolved`.

Set up Configuration Items in Service Desk to Represent Computers in MOM

At this point, new incidents in Service Desk originating from MOM will probably not be associated with any configuration item. The integration tries to match the Computer Name field of the original alert in MOM to the search code of a configuration item in Service Desk, and you may very well not have a configuration item that is set up this way. Therefore, you should change existing configuration items in Service Desk representing computers known to MOM to have their computer name as search code and create new configuration items for the computers not known in Service Desk.

Configuration items can be added or changed by any administrator using the OpenView console. You can assign whatever CI category you want for the computers known to MOM: your choice does not affect the integration. For convenience, consider introducing a CI category for computers known to MOM.

Once you have added configuration items using the MOM computer name as search code in Service Desk, incidents originating from MOM will be associated with them automatically.

Map Impact Values

You need to map MOM severity alert levels to Service Desk incident impact values.

To map severity alert levels to impact values:

1. Log on to the OpenView console using an account with administrator priveleges.
2. In the **OpenView Configuration** workspace group, navigate to **Data> Data Exchange> Import Mapping**.

   Import mappings are displayed in the right panel.
3. Right-click the Mom import mapping, and select **Edit** from the drop-down menu.

   The Mom Import Mapping dialog box opens. The import mapping contains one entity mapping for MOM alarms to Service Desk incidents.
4. Select the entity mapping, then click **Edit**.
5. The Incident Entity Mapping dialog box opens. Attribute mappings are displayed in the Attribute Mappings list.
6. Select the **Impact** attribute mapping, then click **Map**.

   The Attribute Mappings dialog box appears.
7. Build a list of value mappings. To add a value mapping:
   a. In the **External Value** field, type a MOM alert severity level ID.

      For example, if a MOM alert severity value `Critical Error` has the ID `50`, enter **`50`** in the **External Value** field.
   b. In the **Internal Value** field, select the corresponding incident impact value to which the severity level should be mapped.
   c. Click **Add to List**.
   d. When the list of value mappings is complete, click **OK**.
8. Save the import mapping.

Troubleshooting

This section provides hints on how to proceed if you encounter difficulties with the MOM integration.

Is the Forwarding Rule Configured Correctly?

If you experience problems forwarding alerts to Service Desk, start by reverting to the default alert processing rule to exclude the possibility that the integration does not work due to additional limitations you have set.

Verify that the rule group Create Incident in Service Desk is enabled. If it is enabled with default settings, it should work correctly, unless there is a problem with OvObsLoadObject.

The response section of the Create Incident in Servicedesk alert includes a VBS script. The script performs the actual communication steps to Service Desk. Just below the header section in the script is the variable `DebugFlag`. When this variable is set to `TRUE`, a log file called OvSdMom.log is created in the standard log directory. You may find this log file useful for debugging purposes.

Is OvObsLoadObject started correctly?

Once you have established that the forwarding rule is active and working correctly but we still do not see any incidents forwarded, the problem must be in the configuration of `OvObsLoadObject`. There should be an `OvObsLoadObject_error.log` file and possibly an `OvObsLoadobject.log` file in the `%OvInstallDir%\data\log` directory, which in turn should be located in the directory in which you have installed the integration. If there are no logs there, the alert processing rule may be misconfigured or the `LOGFILE` parameter in `OvObsLoadObject.conf` may have been changed. In the latter case, check where the `LOGFILE` parameter points to now.

The `OvObsLoadobject.log` file contains successfully forwarded events, the `OvObsLoadObject_error.log` file contains events that could not be forwarded. In `OvObsLoadObject_error.log`, examine the `SERVER_RESPONSE` and `CLIENT_RESPONSE` fields in every section. `CLIENT_RESPONSE` entries indicate a configuration error in

OvObsLoadObject.conf. When this error occurs, review the installation steps found under “Configure the Object Loader” on page 133. If this is not the cause of the client error, look at the Service Desk management server:

- The SERVER parameter in OvObsLoadObject.conf should match you Service Desk server.
- The application server should have the HTTP listener enabled on the port you specified with the PORT parameter (On standard installations of the management server, it is enabled and listening to port 30980).
- The HTTP listener on the management server should be set up to accept connections from the IP address of your MOM consolidation server (standard installations accept connections from any address).

SERVER_RESPONSE entries that do not contain the value "OK" usually indicate problems with the account password or its application account's roles. By default, the MOM integration account is given the helpdesk role, which should be sufficient unless you have changed or removed this role. The integration account needs to be able to insert incidents and see the corresponding configuration items.

Troubleshooting the Database Rules

The Service Desk database rules invoke a program on the MOM consolidation server (OvSdToMom.pl) to send information to MOM. You can instruct this program to log more information about its activity to find out that it is actually being started and if not, why it is failing. To do this, you must add a parameter to the database rules that are part of the integration. Open the database rules from the OpenView console and add the following parameter to all command exec actions:

```
"-verbose=%OvInstallDir\\data\\log\\sd2mom.log"
```

Make sure there is a space between the end of this parameter and the next one.

If the log file is not being updated after you have updated both database rules to contain this extra parameter, you should check the following conditions and correct them if they are not met:

- Is the Service Desk agent on the MOM consolidation server running at all?

- Is the MOM Integration account's host set to the fully qualified hostname of the MOM consolidation server?

Making Sure Closing an Incident Resolves the Corresponding Alert

The MOM: Close incident database rule is responsible for closing alerts corresponding to incidents whenever the incident is closed. The text of the status code `Closed` can be modified in Service Desk, and the internal representation of the status "Resolved" may change as well. Therefore, two parameters of `OvSdToMom.pl` control how to interpret the status codes at either end.

The parameter `_SD_CLOSE` of `OvSdToMom.pl` represents what the `STATUS` parameter of `OvSdToMom.pl` should be compared to. If the value for `STATUS` and `_SD_CLOSE` match, the `OvSdToMom.pl` program will set the status code of the alert to the value of the parameter `_MOM_CLOSE`. If the text for status code `Closed` in Service Desk is changed or the internal representation of status `Resolved` in MOM changes, you should change the `_MOM_CLOSE` and `_SD_CLOSE` parameters in the MOM: Close Incident database rule to make sure that closing an incident in Service Desk resolves the corresponding alert.

D Microsoft Systems Management Server (SMS) Integration

This appendix describes the Service Desk integration with Microsoft Systems Management Server (SMS).

The SMS integration uses data exchange features to import SMS data into the OpenView database to which your Service Desk management server is connected. SMS supports enterprise configuration management, software distribution and inventory collection.

The SMS integration offers the user the following capabilities:

- For each listed SMS item, a configuration item is created in the OpenView database.
- Relations between the SMS items are also imported in the OpenView database.

Features

The SMS integration allows you to import SMS data using Data Exchange features in Service Desk. SMS provides enterprise configuration management, software distribution and inventory collection. The installation and configuration of the integration is explained in the following sections.

The integration provides a default configuration file for the Data Exchange Exporter.

By default, data is extracted for the following SMS items:

- Site
- Processing Unit
- Operating System
- Hard Disk
- CD ROM
- Floppy
- Software

Default import mappings (including templates) for the data exchange importer are provided with this integration. For each listed SMS item, a configuration item is created in Service Desk. Relations between the SMS items are also imported in the Service Desk database. The following tables show the import mappings.

Table D-1 Site Attribute Mappings

Site	Configuration Item Attribute
SMSSITE	Search code
Site name	Name
Production	Status
Yes	Unique
Remark	Imported from Microsoft SMS

Table D-1 Site Attribute Mappings

Site	Configuration Item Attribute
Hardware	Category

Table D-2 Processing Unit Attribute Mappings

Processing Unit	Configuration Item Attribute
SMSPU	Search code
System name	Name
System role	Name 2
Production	Status
Yes	Unique
Remark	Imported from Microsoft SMS
Hardware	Category
System type	Serial number
SMS ID	Source ID

Table D-3 Operating System Attribute Mappings

Operating System	Configuration Item Attribute
SMSOS	Search code
OS name	Name
SP	Name 2
Production	Status
No	Unique
Remark	Imported from Microsoft SMS
Operating System	Category
OS version	Serial number

Table D-3 Operating System Attribute Mappings

Operating System	Configuration Item Attribute
1,000,000	Max. installations

Table D-4 Hard Disk Attribute Mappings

Hard Disk	Configuration Item Attribute
SMSHARDDISK	Search code
HD name	Name
Capacity	Name 2
Production	Status
No	Unique
Remark	Imported from Microsoft SMS
Hard disk	Category
1,000,000	Max. installations

Table D-5 CD ROM Attribute Mappings

CD ROM	Configuration Item Attribute
SMSCDROM	Search code
CD name	Name
Manufacturer	Name 2
Production	Status
No	Unique
Remark	Imported from Microsoft SMS
CD/DVD Drive	Category
1,000,000	Max. installations

Table D-6 Floppy Attribute Mappings

Floppy	Configuration Item Attribute
SMSFLOPPY	Search code
Floppy name	Name
Production	Status
No	Unique
Remark	Imported from Microsoft SMS
Hardware	Category
1,000,000	Max. installations

Table D-7 Software Attribute Mappings

Software	Configuration Item Attribute
SMSOFTWARE	Search code
Software name	Name
Company	Name 2
Production	Status
No	Unique
Remark	Imported from Microsoft SMS
Software	Category
1,000,000	Max. installations

Installation

This section describes how to install the SMS integration.

Before you start:

1. Ensure you are familiar with the contents of the Service Desk Installation Guide, Chapter 2 “Preparing to Install HP OpenView Service Desk 5.1”, including the section “The HP OpenView Installer” on page 51 – this contains a description of the HP OpenView Installer and its associated screens.
2. See “System Specifications and Requirements” on page 35 of the Service Desk Installation Guide for general information on install prerequisites for HP OpenView Service Desk components.

The SMS integration must be installed on a computer on which a Service Desk client is installed.

To install the SMS integration:

1. Log on to the SMS server using an account with system administrator rights.
2. Run the executable installation file.

   The installation file is located in the Service Desk media root directory. The file is named `smsintegration_<revision>_setup.exe`, where `revision` refers to the build identifier, for example 5.00.722.
3. The **Introduction window** of HP OpenView Installer is displayed. If you previously installed any Service Desk component, the install wizard prompts you to use the install configuration file or decline to use it. For information on this file, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.
4. The HP OpenView Installer starts and guides you through the install process. For a full description of the standard install procedure for Service Desk components, see “Install Screen Overview” on page 55 of the Service Desk Installation Guide.

The following files are installed:

- `<DataDir>\conf\sd\ovsdexchsmsdata.conf`

  Configuration file for the data exchange exporter. This file contains information how to connect to the SMS database. It also describes what data to extract from the SMS database.

- `<DataDir>\conf\sd\ovsdsmsconfigdata.xml`

  Configuration exchange file that must be uploaded into Service Desk. See “Configure an ODBC Connection to the SMS Database” on page 147. This file contains the Service Desk configuration data for the SMS integration.

- `<InstallDir>\newconfig\conf\sd\ovsdexchsmsdata.conf`

  This file is a copy of the original `<DataDir>\conf\sd\ovsdexchsmsdata.conf file`. It can be used to restore the configuration file to its original state.

- `<InstallDir>\newconfig\conf\sd\ovsdsmsconfigdata.xml`

  This file is a copy of the original `<DataDir>\conf\sd\ovsdsmsconfigdata.xml` file. It can be used to restore the configuration file to its original state.

Configuration

This section describes the following configuration tasks:

- Configure a Database Connection to the SMS Database
- Import the Configuration Exchange File
- Adjust the Service Desk Settings
- Configure Data Source Connection

Configure a Database Connection to the SMS Database

Before you can export data from the SMS database, you must configure a database connection to the SMS database. You can choose what type of connection you want to use. Service Desk supports both ODBC and JDBC connections. Choose which type of connection you will use.

Configure an ODBC Connection to the SMS Database

The specifics of configuring an ODBC data source for a particular database system are described in that database system's documentation. This section provides an example for a SQL server data source. The ODBC data source must be defined on the computer that is running the SMS integration. The SQL server database containing the SMS database tables may be located on any machine, but it is recommended to run the SMS integration on the SMS server. To configure the ODBC data source, install the appropriate driver. Drivers are normally provided with the database system, or may be obtained separately from the database system vendor..

To configure the ODBC connection to a SQL server data source:

1. Open the ODBC Data Source Administrator and navigate to the System DSN tab.
2. Click **Add**, select the SQL Server Driver, then click **Finish**.

   This invokes a wizard which leads you through the process of defining the ODBC data source.
3. In the **Name** field, enter a name for the data source definition.

During the configuration of Service Desk you supply this name in the DSN section.

4. The **Description** field is free text. Enter a description that denotes the purpose or use of the data source definition.
5. In the **Server** field, select the SQL Server hosting the SMS data from the **Server** drop-down list. Click **Next** to continue.
6. Choose which authentication mechanism you want to use. Specify a login that has read access to the SMS database. Click **Next** to continue.
7. Select or type the name of the desired SQL Server database into the **Change the default database to** field.
8. Set the remaining controls in this panel according to the requirements of the selected database. (See your SQL Server administrator for the necessary information.) Click **Next** to proceed.
9. Set the controls in the next panel according to the requirements of the selected database (See your SQL Server administrator for the necessary information.) Click **Finish** to proceed.
10. Click **Test Data Source** to perform a test of your ODBC data source definition. If you have correctly entered the necessary information to define the ODBC data source, the system displays a message confirming that the test completed successfully.
11. Click **OK** to save the data source definition and close the dialog box.

    This concludes the configuration of the ODBC connection.

Configure a JDBC Connection to the SMS Database

You need to create a database user in the SQL Server database that contains the SMS data. The credentials of this user are needed when configuring the data source connection described later in this chapter.

To create a SQL Server database user:

1. On the SMS server, start the Enterprise Manager of SQL Server.
2. Navigate to the SMS database and select the **Users** leaf.
3. Right-click in the right panel and select **New Database User**. Alternatively, from the **Actions** menu, select **New Database User** to open the Database User Properties settings.

4. In the **Login name** field, select <new> from the drop down list.
5. On the General tab, supply a **Name**. Select **SQL Server Authentication**. Supply a **Password**. Specify the SMS Database.
6. On the Database Access tab, specify the SMS database this user has permission to access. Give the user the role of `db_datareader`.
7. Click **OK** to save the data. Confirm the new password of the newly created user.
8. Close the Database User Properties settings.

This concludes the creation of the database user needed for the JDBC connection.

Import the Configuration Exchange File

Upload the configuration data stored in the configuration exchange file `<DataDir>\conf\sd\ovsdsmsconfigdata.xml`.

To import the configuration exchange file:

1. Log on to the OpenView Console as an administrator.
2. Select **File>Configuration Exchange>Import**.

   The Configuration-Exchange Import Wizard opens.
3. Click **Next** on the first page of the import wizard, then click **Add**, then browse to the directory containing the `ovsdsmsconfigdata.xml` configuration exchange file.
4. Select this file and click **Start Import**.
5. After importing the configuration exchange file, exit the import wizard.

The uploaded configuration information comprises the following configuration exchange filter groups:

- SMS Data Exchange Task
- SMS Filter Groups
- SMS Filter
- SMS Import Mappings
- SMS Task Group

- SMS Templates for CI

The Configuration Exchange Filter Group SMS_config_data contains the filters. You can use this filter group to export all SMS configuration data stored in Service Desk to a file. This enables you to transfer your configuration data from one Service Desk deployment (for example, your test environment) to another Service Desk deployment (for example, your production environment).

Adjust the Service Desk Settings

The data to be imported from SMS into Service Desk might contain the same value for the search code of the configuration item. By default, the search code of configuration items is unique. You need to change the Service Desk setting of the search code of configuration items to be non-unique.

To configure non-unique configuration item search codes:

1. Log on to the OpenView Console as an administrator.
2. In the **OV Configuration** workspaces, navigate to **Data>Unique Fields**.
3. Double-click **Configuration Item**.
4. In the Unique Fields - Configuration Item window, clear the **Search code** field.
5. Save your changes and close the window.

Configure Data Source Connection

Modify the connection details in the `<DataDir>\conf\sd\ovsdexchsmsdata.conf` file. You need to change the [DSN] section or the [CONNECTION] and [JDBC] sections in this file depending on the connection type you have chosen. Only one section may be present. Comment out the connection you will use and comment in the connection you will not use. The DSN section in this file contains the ODBC connection details.

To configure a JDBC or an ODBC connection:

1. Open the `ovsdexchsmsdata.conf file` in a text editor.

2. Modify the NAME, USR and PWD entries to reflect your environment.

   This information was specified when you created the ODBC connection as described in “Configure an ODBC Connection to the SMS Database” on page 147.

   [DSN]

   NAME=<**ODBC System DSN**>

   USR=<**user name**>

   PWD=<**user password**>

3. Save the file and exit your text editor.

To configure the JDBC connection:

1. Open the `ovsdexchsmsdata.conf` file in a text editor.
2. Modify the URL, USR and PWD entries to reflect your environment.

   This information was specified when you created the SQL Server database user as described in “Configure a JDBC Connection to the SMS Database” on page 148.

   [JDBC]

   URL=jdbc:inetdae://<**SQL Server hostname**>:1433?database=<**SQL Server DB name**>

   USR=<**user name**>

   PWD=<**user password**>

3. Save the file and exit your text editor.

This concludes the configuration of Service Desk.

Import SMS Data into Service Desk

You can import SMS data into Service Desk from the OpenView console on the computer on which the SMS integration software is installed. A data exchange task group in Service Desk performs the export of the data out of SMS and the import of the data into Service Desk within the same job.

To import SMS data into Service Desk:

1. Log on to the OpenView Console as an administrator.
2. In the **OV Configuration** workspaces, navigate to **Data>Data Exchange>Task Group**.
3. Right-click **Exchange SMS data Task Group** and select **Start** from the drop-down menu.
4. Provide the password of the user importing the SMS data.
5. Click **OK** to continue.
6. Click **OK** to start the export and import process.

Troubleshooting

This section provides hints on how to proceed if you encounter difficulties with the SMS integration.

In case of problems, always check the log files. The default log file of the export process is `<DataDir>\data\log\ovsdexchsmsdata.log`. The default log file of the import process is located at: `C:\Documents and Settings\<User Name>\Application Data\HP OpenView\data\data_exchange\log`.

java.net.ConnectionException: Connection refused: connect

Symptom

When the SMS Exchange Task runs, a window pops up with the error message: `java.net.ConnectionException: Connection refused: connect`.

Possible Cause

ODBC connection is not set up correctly, or the connection details in the `<DataDir>\conf\sd\ovsdexchsmsdata.conf` configuration file are wrong.

Solution

Correct the ODBC connection or correct the DSN, USR and PWD entries in the configuration file.

Error running import/export ... Failed to connect to server

Symptom

When the SMS Exchange Task runs, a window pops up with the error message: `Error running import/export … Failed to connect to server`.

Possible Cause

The wrong password for the user who imports the SMS data has been supplied.

Solution

Provide the correct password for the importing user.

Data Exchange - Fatal Error

Symptom

When the SMS Exchange Task runs, a window pops up with the error message: `Data Exchange - Fatal Error`.

Possible Cause

The user specified in the ODBC connection and in the `<DataDir>\conf\sd\ovsdexchsmsdata.conf` file does not have enough access rights to the SMS tables.

Solution

In SQL Server, make sure the specified user has enough access rights to the SMS tables. Grant the user access to the correct database and permit the user the role of `db_datareader`.

E Examples

The following examples demonstrate how to use data exchange to extract information from external data sources and import it into the OpenView database connected to a Service Desk management server.

Importing From a Spreadsheet

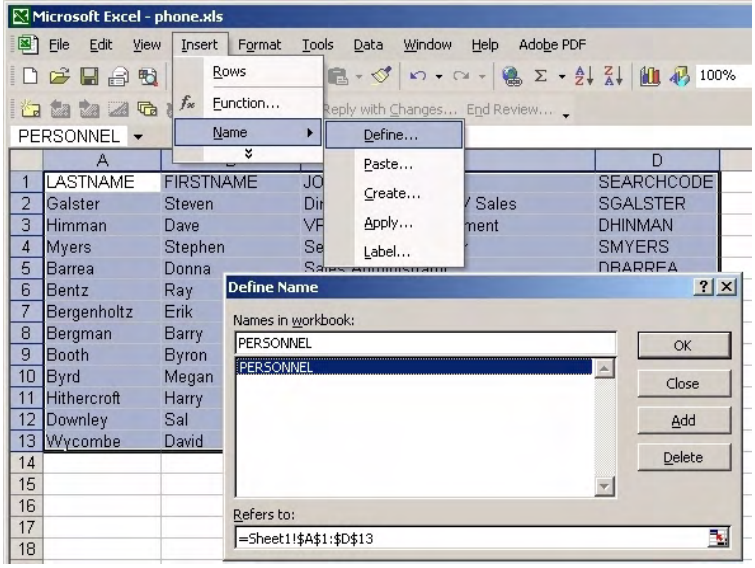
This example demonstrates how to import data from a Microsoft Excel spreadsheet. The example assumes that the data source is created using the Microsoft Office XP version of Excel.

Step 1. Create and prepare the Excel spreadsheet.

1. Open a new spreadsheet in Microsoft Excel.
2. Provide column headings. In this example, `LASTNAME`, `FIRSTNAME`, `JOBTITLE`, and `SEARCHCODE` are used for column headings.
3. Add a few rows of personnel information below the column headings.
4. Select all data from the Excel sheet, including the column headings.
5. Specify a name for the selection. The data selected for this example is given the name `PERSONNEL`.

   To define an area in the spreadsheet, click **Insert→Name→Define** from the menu bar. You can then view, add, and delete the names that define each data area as shown in Figure E-1.

Figure E-1 Prepare the Excel Spreadsheet

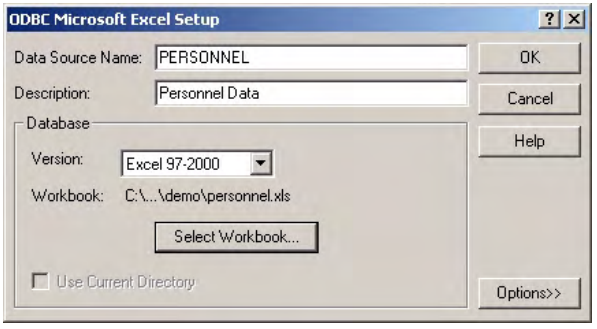


6. Save the spreadsheet under the filename `personnel.xls`.

Step 2. Create an ODBC data source for the Excel spreadsheet.

1. Start ODBC Data Source Administrator from the Windows Control Panel.
2. Select the **System DSN** tab and click **Add**.
3. Select the driver you want to use. `Microsoft Excel Driver (*.xls) version 4.00.6019.00` is used in this example. Then click **Finish**.
4. In the ODBC Microsoft Excel Setup dialog box, provide a **Data Source Name** and a **Description**, then click the **Select Workbook** button and browse to the spreadsheet you selected in Step 1. Click **OK** to save the ODBC data source.

Figure E-2 ODBC Microsoft Excel Setup Dialog Box



Step 3. Configure the extractor.

1. Open a new file in a text editor and insert the following text:

```
[DSN]
NAME=PERSONNEL
USR=
PWD=

[SYSTEM]
LOG=TRUE
XML=TRUE
LOG_FILE=C:\data_exchange\log\personnel.log
OUTPUT_FILE=C:\data_exchange\xls\personnel.xls
XML_OUTPUT_FILE=C:\data_exchange\xml\personnel.xml
```

```
APPLICATION_NAME=PERSONNEL
ENCODING=ISO-8859-1

[CLASSES]
NAME=CLASSPERSONNEL

[CLASSPERSONNEL]
SOURCE=PERSONNEL
ATT=[LASTNAME],[FIRSTNAME],[JOBTITLE],[SEARCHCODE]
COLUMNS=[LASTNAME],[FIRSTNAME],[JOBTITLE],[SEARCHCODE]
LOADTABLE=TRUE
```

You may need to modify the extractor configuration file so that it matches your environment.

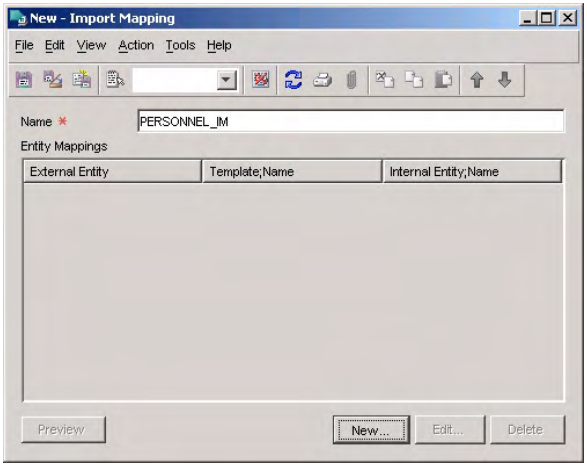
2. Save the extractor configuration file under the name personnel.ini.

   For further information about the keywords and syntax used in extractor configuration files, see “Data Extraction Configuration File” on page 22.

Step 4. Create an import mapping.

1. In the **OV Configuration** workspace group, expand the **Data** →**Data Exchange**→**Import Mapping** branch.
2. Right-click in the right panel and select **New Import Mapping**. The New Import Mapping dialog box appears:

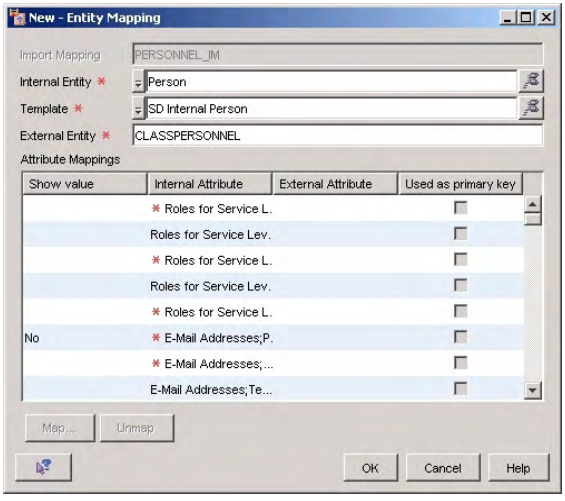
Figure E-3 New Import Mapping Dialog Box



3. Give the import mapping a name. For this example, it is PERSONNEL_IM.

4. Click the **New** button to create a mapping from a particular external class to a specific object type. The New Entity Mapping dialog box appears:

Figure E-4 New Entity Mapping Dialog Box



In this example, you only need to create a mapping for the CLASSPERSONNEL class.

5. In the **External Entity** field, type the name of the class to be mapped.

6. In the **Internal Entity** field, specify the object type in the OpenView database to which the external class should be mapped.

   The mapping you create is dependent on the type of information you are importing. Because the CLASSPERSONNEL class contains personnel information, you should map it to the Person object type.

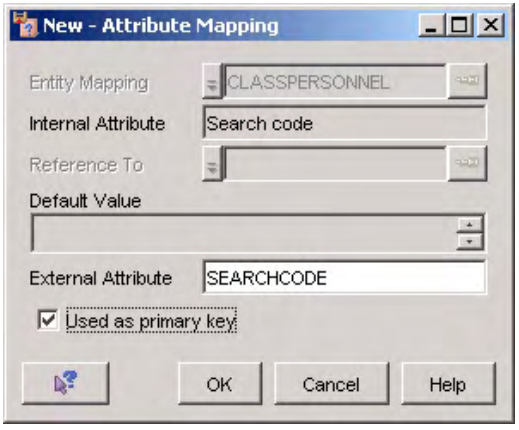
7. In the **Template** field, specify the template to be used for the creation of the person objects.

   In this example, the SD Internal Person template is used.

8. In the **Attribute Mappings** list, specify how each external attribute should be mapped to an attribute of the Person object type.

In this example, the SEARCHCODE external attribute should be mapped to the Search Code attribute of the Person object type. Click the **Search Code** attribute in the **Attribute Mappings** list, then click the **Map** button. The New Attribute Mapping dialog box appears:

Figure E-5 Mapping Attributes



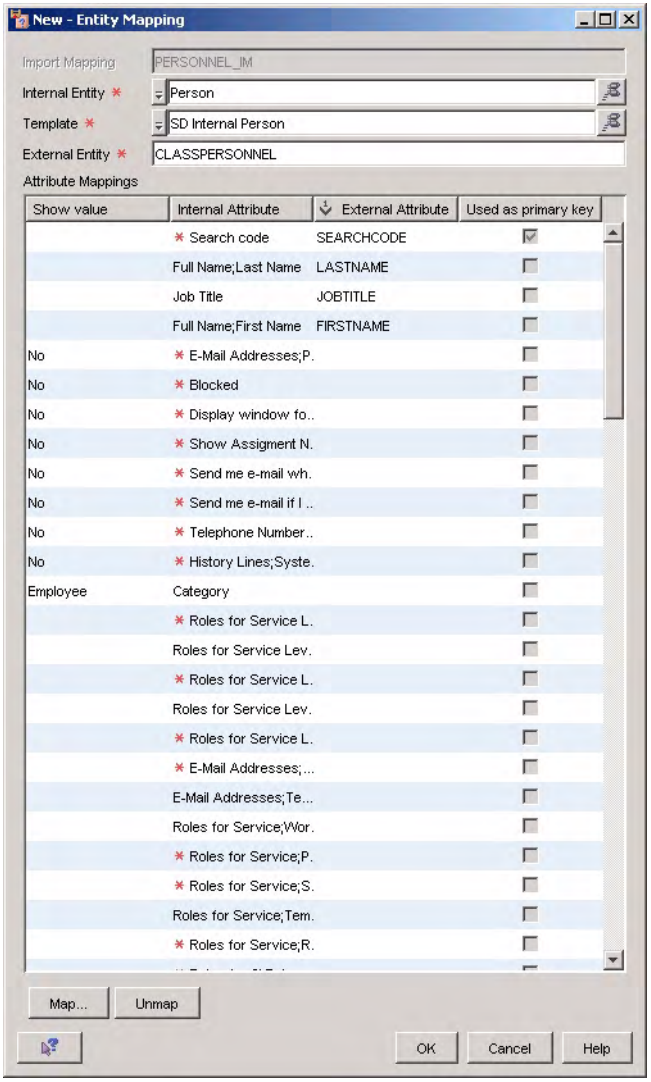
9. In the **External Attribute** field, type the name of the external attribute that should be mapped to the attribute you selected.
10. Select the **Used As Primary Key** check box to indicate that data exchange should use this attribute to identify instances of this object type.

NOTE You must designate at least one unique attribute as the binding key. This is usually the search code or another unique key.

11. Click **OK** to save your mapping and return to the Entity Mapping dialog box.

12. Provide mappings to the other internal attributes in the same way. The following dialog box shows the completed import mapping for this example:

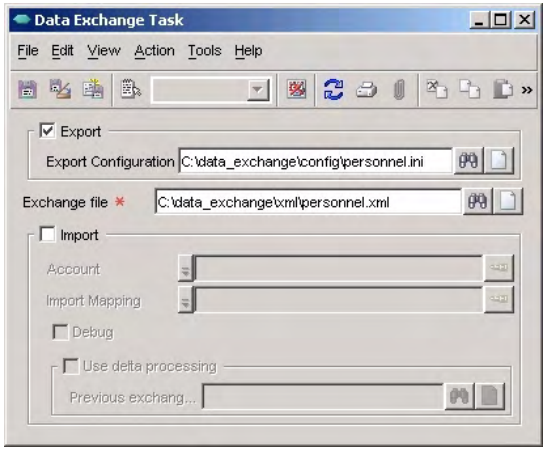
Figure E-6 Complete Import Mapping



Step 5. Export the Excel spreadsheet data.

1. In the **OV Configuration** workspace group, expand the **Data →Data Exchange→Data Exchange Task** branch. Right-click in the Data Exchange Task window and select **New Data Exchange Task**. The New Data Exchange Task dialog box appears:

Figure E-7 Example Export Data Task



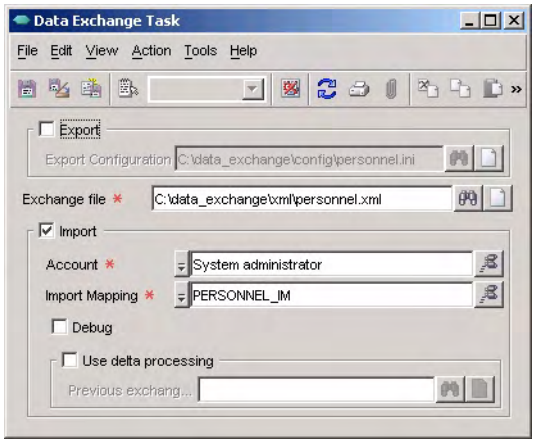
2. Select the **Export** check box.
3. In the **Export Configuration** field, specify the name of the extractor configuration file. For this example, it is `personnel.ini`. Use the **Quick Find** button to the right of the field to find the configuration file.
4. In the **Exchange file** field, specify the name of the XML file as specified in the extractor configuration file. For this example, the XML file is called `personnel.xml`.
5. Click **File→Save** to save this task for use another time.
6. Click **Action→Start** to export the data from the Excel file.
7. Check the contents of the `personnel_exp.log` file to verify that the export was successful.

Step 6. Import the data from the exchange file.

1. In the **OV Configuration** workspace group, expand the **Data →Data Exchange→Data Exchange Task** branch. Right-click in the Data Exchange Task window and select **New Data Exchange Task** if you

want to create a new task for the import process. Alternatively, you can edit the task created in the previous step. The Data Exchange Task dialog box appears:

Figure E-8 Example Import Data Task



2. In the **Exchange file** field, specify the name of the exchange file that contains the data to be imported. In this example, the file name is `personnel.xml`.
3. Select the **Import** check box.
4. Specify the account you use for data exchange purposes.
5. In the **Import mapping** field, specify the import mapping you created for importing the Excel file. In this example it is `PERSONNEL_IM`. Use the **Quick Find** button to the right of the field to find the import mapping.
6. Select the **Debug** check box and click **Action→Start** to start the import process.

   The Password dialog box appears:

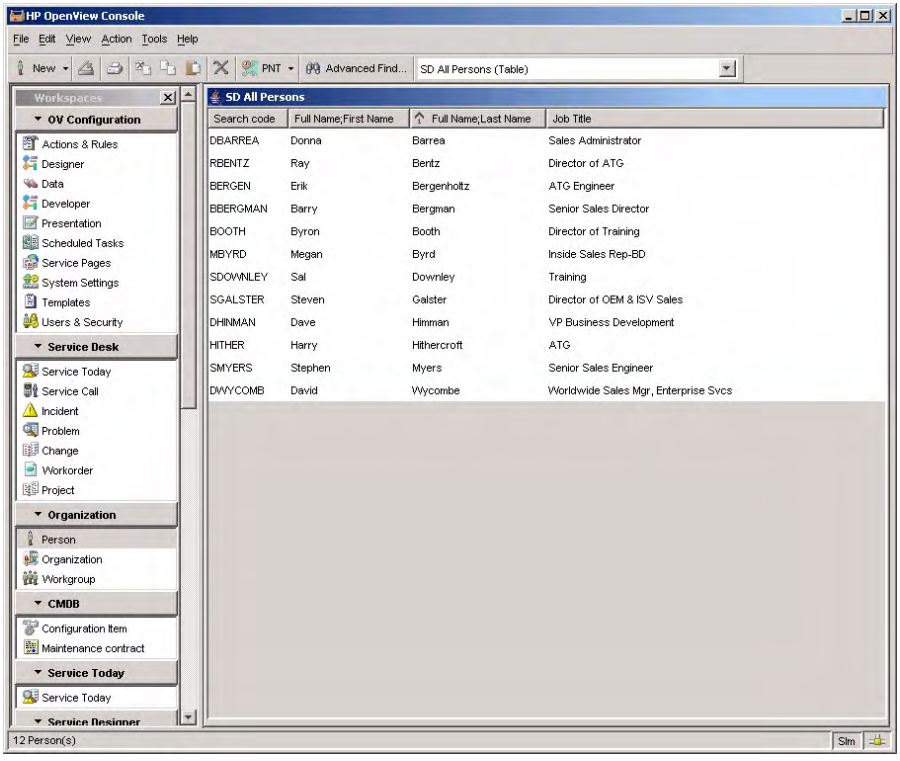
Figure E-9 Password Dialog Box

7. Type the password associated with the account specified in Step 4, then click **OK**.

8. Check the contents of the `PERSONNEL_IM_imp.log` file to verify that the import was successful. You should also verify that the information is visible in the OpenView console.

   Figure E-10 displays the imported information in the OpenView console:

Figure E-10 Imported Information Displayed in the OpenView Console



Importing From a Database

This example demonstrates how to use a JDBC connection to import data from an Oracle database. The data values differ from those imported from the spreadsheet in the previous example, but the format is identical. This enables the use of the same import mapping (`PERSONNEL_IM`) created in the previous example.

Step 1. Prepare the database.

In this example, the data is inserted into a new table named `PERSONNEL` in the same Oracle database to which the Service Desk management server is connected.

1. Create a SQL script with the following contents, and save it under the filename `orademo.sql`:

```
CREATE TABLE PERSONNEL(

    LASTNAME VARCHAR(50) NOT NULL,

    FIRSTNAME VARCHAR(40) NOT NULL,

    JOBTITLE VARCHAR(100) NOT NULL,

    SEARCHCODE VARCHAR(20) NOT NULL PRIMARY KEY );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Foster', 'Paul', 'Director of Communication Resources', 'PFOSTER' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Stapleton', 'Nigel', 'Quality Assurance Director', 'NSTAPLETON' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Walters', 'Elizabeth', 'Vice President', 'EWALTERS' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Bacon', 'Malcolm', 'Regional Sales Director', 'MBACON' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Hanrahan', 'Theresa', 'Systems Recovery Specialist', 'THANRAHAN' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES( 'O''
Brien', 'Conor', 'Systems Administrator', 'COBRIEN' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Kelly', 'Stephen', 'Line Manager', 'SKELLY' );
```

```
INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Ford', 'Brad', 'Operations Instructor', 'BFORD' );

INSERT INTO PERSONNEL( LASTNAME, FIRSTNAME, JOBTITLE, SEARCHCODE ) VALUES(
'Thorpe', 'Caroline', 'Chief Technology Officer', 'CTHORPE' );

COMMIT;
```

2. Run the following command from a command prompt to execute the script:

   ```
   sqlplus <user>/<password> @orademo
   ```

   In this example, the database user ID is `slm_ID`, and the password is `slm_pw`:

   ```
   sqlplus slm_ID/slm_pw @orademo
   ```

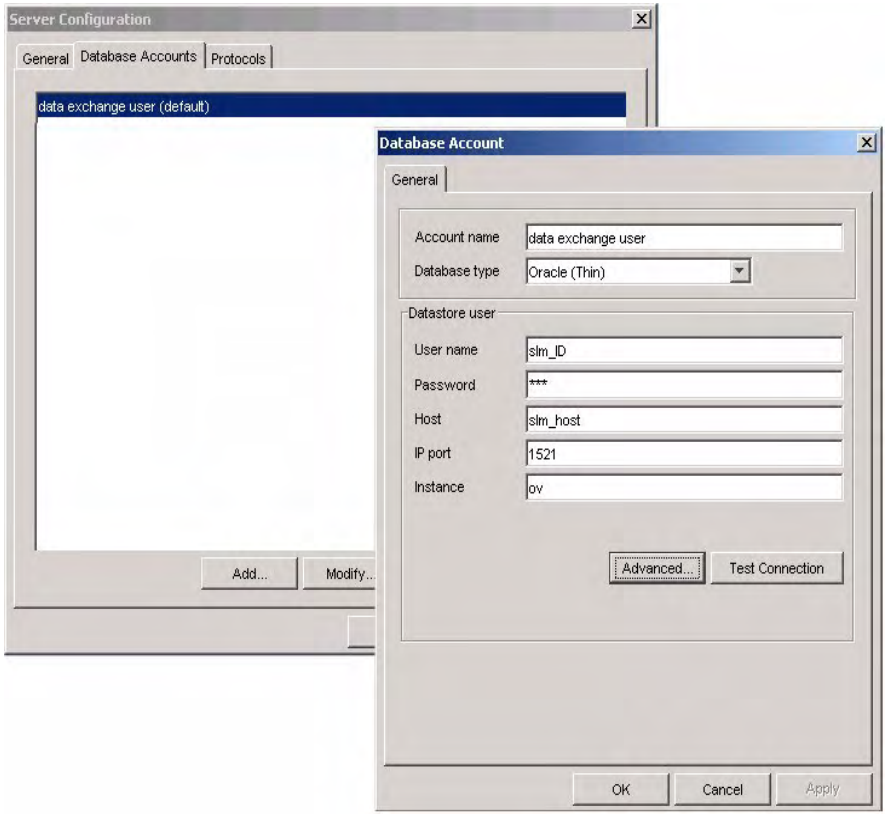
   The script creates a table named `PERSONNEL` with columns named `LASTNAME`, `FIRSTNAME`, `JOBTITLE`, and `SEARCHCODE`., and populates the table rows with the values specified in the script.

Step 2. Configure the extractor.

1. Open the sample extractor configuration file created in the previous example: `personnel.ini`.
2. Copy the file to a new .ini file. In this example the new .ini file is named `personnel_jdbc.ini`.
3. Modify the extractor configuration file so that it matches your environment. The example extractor configuration file makes the following assumptions about the database account:
   - The database is installed on a server with the hostname `slm_host`
   - The database account user ID is `slm_ID`
   - The password for the database user account is `slm_pw`
   - The name of the database instance is `ov`
   - The IP port is `1521`

You should substitute the appropriate values according to your own database account settings displayed in the server settings editor:

Figure E-11 Database Account Settings




```
[CONNECTION]
TYPE=JDBC

[JDBC]
NAME=Personnel JDBC
USR=slm_ID
PWD=slm_pw
DRIVER=oracle.jdbc.driver.OracleDriver
URL=jdbc:oracle:thin:@slm_host:1521:ov

[SYSTEM]
LOG=TRUE
XML=TRUE
```

```
DUMP=FALSE
TXT=FALSE
LOG_FILE=C:\data_exchange\log\personnel_jdbc.log
XML_OUTPUT_FILE=C:\data_exchange\xml\personnel_jdbc.xml
APPLICATION_NAME=Personnel JDBC

[CLASSES]
NAME=CLASSPERSONNEL

[CLASSPERSONNEL]
SOURCE=PERSONNEL
ATT=[LASTNAME],[FIRSTNAME],[JOBTITLE],[SEARCHCODE]
COLUMNS=[LASTNAME],[FIRSTNAME],[JOBTITLE],[SEARCHCODE]
LOADTABLE=TRUE
```

The `CLASSES` section of the extractor configuration file is unchanged from the previous example because the table and column names in the data source are identical. This means that the source data is extracted to an XML file with the same structure as in the previous example, enabling the use of the same import mapping as before (`PERSONNEL_IM`).

For further information about the keywords and syntax used in extractor configuration files, see “Data Extraction Configuration File” on page 22.

Step 3. Export the database data.

You export the data in the same was as in the previous example, by creating a data exchange task. Specify the correct extractor configuration file. For this example, it is `personnel_jdbc.ini`. Export the data to an exchange file named `personnel_jdbc.xml`.

Step 4. Import the data from the exchange file.

You import the data in the same was as in the previous example, by creating a data exchange task. Specify the correct exchange file that contains the data to be imported. In this example, the file name is `personnel_jdbc.xml`.

Use the import mapping created in the previous example: `PERSONNEL_IM`.

Importing Data With Relations

This example focuses on the process of configuring an extractor file and creating an import mapping to import data from a data source. Special attention is given in this example to how you import relations. The information about adjusting the configuration file and creating the import mapping is combined for each class in this example so that you can directly see the relationship between the two. The overall process contains the following ordered steps:

1. Analyze your data source to determine what to export and how to store it.
2. Create an ODBC link
3. Configure the extractor
4. Export data
5. Map the exported data to templates, objects and attributes in the OpenView database.
6. Import the data.

Defining Relations in the Configuration File

The way you configure the extractor configuration file affects the way relations are imported. The following types of relations can be imported:

- N:1 Relations Based on a Search Key
- N:N (Parent-Child) Relations
- Relations of Type 1:Rel:1

The following sections explain the relation types in more detail, including information on how to configure the extractor configuration file so that related objects are successfully imported into the OpenView database.

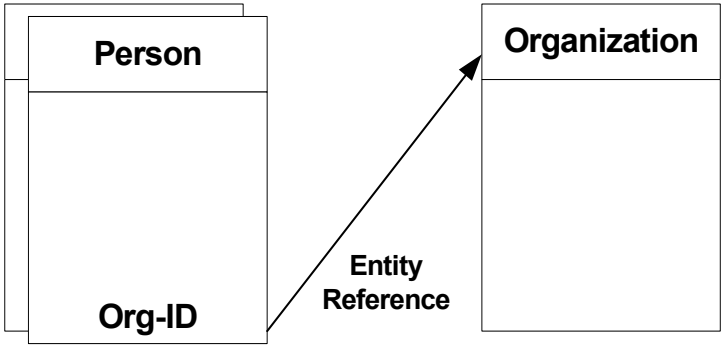
The Service Desk data dictionary provides a description of all object types and attributes in the OpenView database including the relations between objects. If the attribute for an object has a 1:1 relation (for example, between the Caller attribute in a service call and a Person object), the entity reference to the other object type is shown. If the

attribute has a 1:N relation (for example, between the History Lines attribute in a service call and the entity set of History Line objects), the entity set reference is shown.

N:1 Relations Based on a Search Key

In an N:1 relation, the N refers to one or more objects related to one other object. This type of relation uses a search code as an entity reference to determine the relationship between the objects. When configuring the extractor configuration file, you need to make sure that the object that is referenced using a search code is exported first. The order is determined by how the classes are listed in the [CLASSES] section. If the referenced entity is not exported to the XML source file first, an error results when importing. An example of this type of relation is the representation of the members of an organization. Figure E-12 shows this relation between the Member of Organization attribute of the Person object type referencing the Organization object type:

Figure E-12 Example N:1 Relation



Following is a portion of a configuration file demonstrating how to export this type of relation. Note that the ORGANIZATION class is configured to be exported before the PERSON class:

```
[ORGANIZATION]
SOURCE=[ORG]
ATT=[SEARCHCODE], [NAME]
COLUMNS=DISTINCT [SEARCHCODE], [NAME]
LOADTABLE=TRUE

[PERSON]
SOURCE=[EMPLOYEE]
ATT=[FIRSTNAME], [LASTNAME], [ORG], [SEARCHCODE2]
```

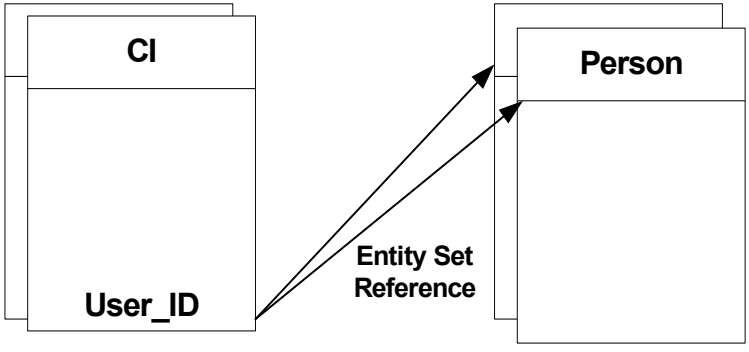
```
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [ORG],
[EMPLOYEE].[SEARCHCODE] AS [SEARCHCODE2]
LOADTABLE=TRUE
```

See “Configuring the ORGANIZATION Class” on page 184 for a description of the ORGANIZATION class. To complete the process of importing this relation, the import mapping you create for the PERSON class needs to use a search code as a reference to the Organization object. “Import Mapping for the PERSON Class” on page 186 describes the import mapping for this example.

N:N (Parent-Child) Relations

This type of relation associates one or more objects in the OpenView database to one or more other objects. This type of relation uses multiple search codes or an entity set reference to make the relationship. The extractor configuration file is modified differently for an N:N relation than an N:1 relation. For the N:N relation, you can use PARENT, PARENT_RELATION, and PARENT_RELATION_NAME fields to specify the parent-child relation. An example of this type of relation is the representation of the user of a configuration item. Figure E-13 illustrates this relation with multiple persons using multiple configuration items. Configuration item users are related to Person objects using an entity set reference. The relationship can be made in either direction:

Figure E-13 Example N:N Relation



Following is a portion of an extractor configuration file demonstrating how to export this type of relation. In this example, PERSON is the parent class, and EQUIPMENT is the child class:

```
[PERSON]
SOURCE=[EMPLOYEE]
ATT=[FIRSTNAME], [LASTNAME], [ORG], [SEARCHCODE2]
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [ORG],
[EMPLOYEE].[SEARCHCODE] AS [SEARCHCODE2]
LOADTABLE=TRUE

[EQUIPMENT]
SOURCE=[EQUIPMENT]
ATT=[NAME], [USER], [SEARCHCODE3], [TYPE]
COLUMNS=DISTINCT [NAME], [USER], [EQUIPMENT].[SEARCHCODE] AS
[SEARCHCODE3], [TYPE]
LOADTABLE=TRUE
PARENT=PERSON
PARENT_RELATION=[USER]=[SEARCHCODE2]
PARENT_RELATION_NAME=USER
```

To complete the process of importing the configuration item user relation, the import mapping for the child class must include a search code as a reference to the parent class. Refer to “Configuring the EQUIPMENT Class” on page 189 for the description of the child class in this example. Refer to “Import Mapping for the EQUIPMENT Class” on page 190 for a description of the accompanying import mapping.

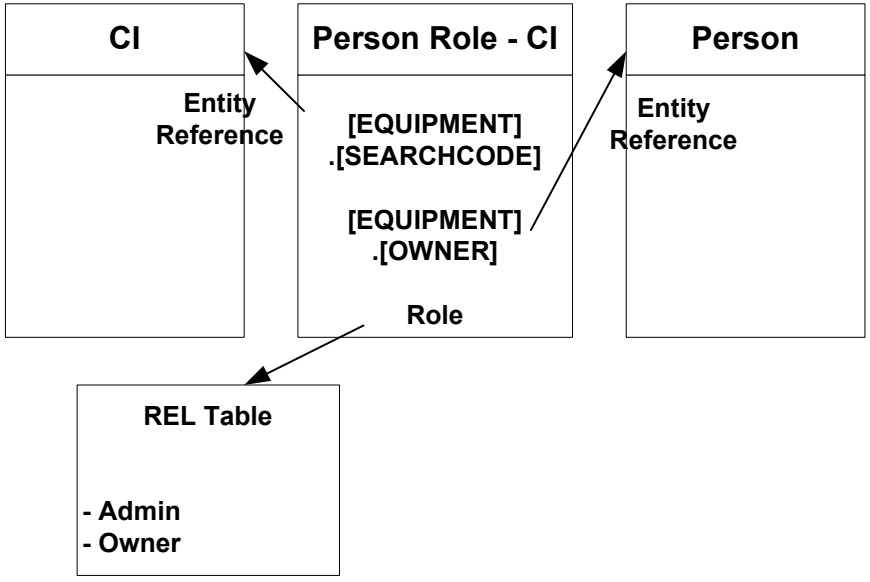
As an alternative approach, you can choose to handle N:N relations in the same way as 1:Rel:1 relations (see “Relations of Type 1:Rel:1” on page 172 for further information about this relation type). The remainder of this example demonstrates how to handle N:N relations for configuration item users by adding PARENT, PARENT_RELATION, and PARENT_RELATION_NAME statements in the extractor configuration file and specifying a suitable import mapping. This approach provides better performance especially when importing large quantities of data, but requires more memory. For information on how to modify the extractor configuration file and import mapping to import N:N relations in the same way as 1:Rel:1 relations, see “Importing N:N Relations as 1:Rel:1 Relations” on page 195.

Relations of Type 1:Rel:1

The 1:Rel:1 relation type is specified by creating an additional class for the purpose of describing the relation between two or more classes. You can use an entity reference with a search code and an entity reference to a relation table in the OpenView database to describe the relation. When using a reference to a relation table, the relation information is maintained in a separate relation code table. You can map objects to the

values in the relation table to define the type of relation. You can also create value mappings for the relations. An example of this type of relation is the representation of the owner of a configuration item. Figure E-14 illustrates this relation.

Figure E-14 Example 1:Rel:1 Relation



This relation associates a configuration item object to a person object. Person Role - CI is the intermediate object type in the OpenView database. As well as connecting a configuration item to a person, it holds information about the role (in this case, the appropriate role is Owner). Following is a portion of an extractor configuration file demonstrating how to export this type of relation. The OWNERS class exports the information relating configuration items to persons registered as owners:

```
[PERSON]
SOURCE=[EMPLOYEE]
ATT=[FIRSTNAME], [LASTNAME], [ORG], [SEARCHCODE2]
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [ORG],
[EMPLOYEE].[SEARCHCODE] AS [SEARCHCODE2]
LOADTABLE=TRUE

[EQUIPMENT]
SOURCE=[EQUIPMENT]
ATT=[NAME], [USER], [SEARCHCODE3], [TYPE]
```

```
COLUMNS=DISTINCT [NAME], [USER], [EQUIPMENT].[SEARCHCODE] AS
[SEARCHCODE3], [TYPE]
LOADTABLE=TRUE

[OWNERS]
SOURCE=[EQUIPMENT]
ATT=[EQUIP_SEARCHCODE], [OWNER]
COLUMNS=DISTINCT [EQUIPMENT].[SEARCHCODE] AS
[EQUIP_SEARCHCODE], [OWNER]
LOADTABLE=TRUE
```

To complete the process for importing this relation you need to create the import mapping to use a search code as a reference to the object. Refer to Figure E-33 for the complete import mapping.

Data Source

To configure the extractor, you need to know how information is stored in your data source and what you want to export. Many applications provide a table description or similar document that can be used for this purpose. In Service Desk, the Data Dictionary provides this type of information. The Service Desk Data Dictionary is useful for understanding the relations between objects in Service Desk, and for deciding where to import the data in the OpenView database.

The data used for this example is taken from a Microsoft Excel spreadsheet. The major differences between importing data from an Excel spreadsheet and a database is in how the ODBC link is made. See “Preparing the Excel Sheet for Data Exchange” on page 176 and “Defining the ODBC Link” on page 176. When looking at your data, you should always think in terms of the SOURCE or database table, and

COLUMNS within a table. In the Excel spreadsheet, the information is also grouped into tables (that is, SOURCES), and columns, (that is, COLUMNS).

Figure E-15 ORG SOURCE

ORG	
SEARCHCODE	NAME
ORG20001	HPHOLLAND
ORG20002	HPFRANCE
ORG20003	HPUK
ORG20004	HPGERMANY
ORG20005	HPSPAIN

Figure E-16 EMPLOYEE SOURCE

EMPLOYEE				
SEARCHCODE	FIRSTNAME	LASTNAME	ORG	NUMBER
EMP100003	Maarten	van Dongen	HPHOLLAND	020 123456
EMP100004	John	Smith	HPFRANCE	020 234567
EMP100005	Shelley	Long	HPHOLLAND	020 345678
EMP100006	Guy	François	HPSPAIN	020 456789
EMP100007	Helen	Morris	HPUK	020 5678990
EMP100008	Monica	van Velden	HPFRANCE	020 678901
EMP100009	Frank	Zappa	HPGERMANY	020 789012
EMP100010	Joe	Barbarese	HPHOLLAND	020 890123
EMP100011	Pete	McPherson	HPSPAIN	020 901234
EMP100012	Sandra	Berke	HPGERMANY	020 012345
EMP100013	Michael	Sasek	HPSPAIN	020 987654
EMP100014	Zem	Philips	HPSPAIN	020 876543

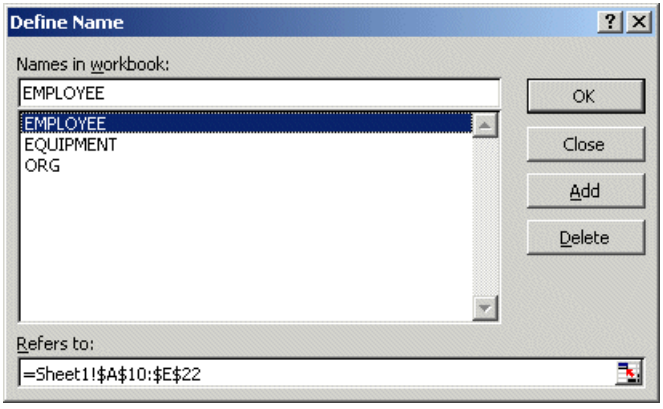
Figure E-17 EQUIPMENT SOURCE

EQUIPMENT				
SEARCHCODE	NAME	USER	OWNER	TYPE
ORGPC001	ORGPC1	EMP100005	EMP100003	PC
ORGPC002	ORGPC2	EMP100004	EMP100003	PC
ORGPC003	ORGPC3	EMP100006	EMP100003	PC
ORGPC004	ORGPC4	EMP100014	EMP100003	PC
ORGPC005	ORGPC5	EMP100012	EMP100004	SERVER
ORGPC006	ORGPC6	EMP100012	EMP100004	SERVER
ORGPC007	ORGPC7	EMP100003		SERVER

Preparing the Excel Sheet for Data Exchange

You need to define the areas of data you intend to export from an Excel spreadsheet. In essence, you group the data into tables that are easier to identify when defining the import mapping. In this example, the following areas are defined in the Excel sheet: EMPLOYEE, ORG, and EQUIPMENT. To define an area in an Excel sheet, click **Insert→Name→Define** from the menu bar. You can then view, add, and delete the names that define each data area as shown in the following example:

Figure E-18 Excel File Name Definition



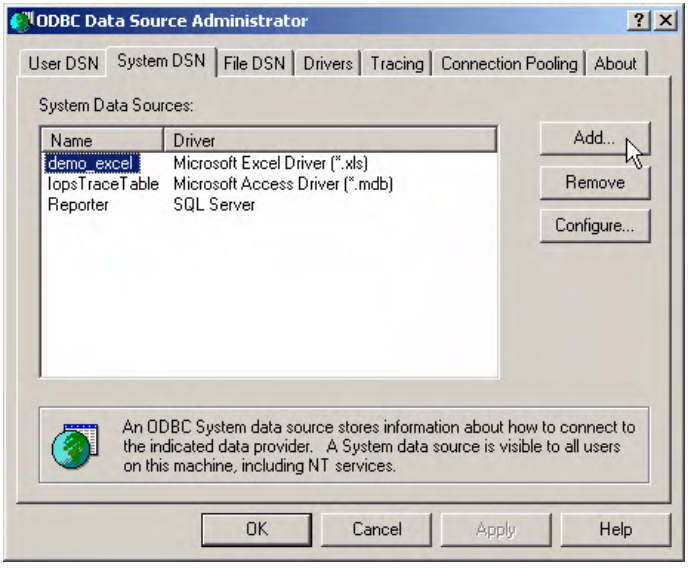
After you create the ODBC link and create or modify the extractor configuration file, you can export the data using a data exchange task. Clear the option for importing data until after you have had time to verify that your export is successful. For more information see “Create a Data Exchange Task” on page 42.

Defining the ODBC Link

An ODBC connection must be configured on the application server where the data you want to import is located. For example, if the exported XML file is on your Service Desk management server, that is where you need to configure the ODBC link from. Service Desk uses System DSN for data exchange. Following is an example of how to set up an ODBC link for Microsoft Excel files. For other types of links, see Table 1-1 on page 23.

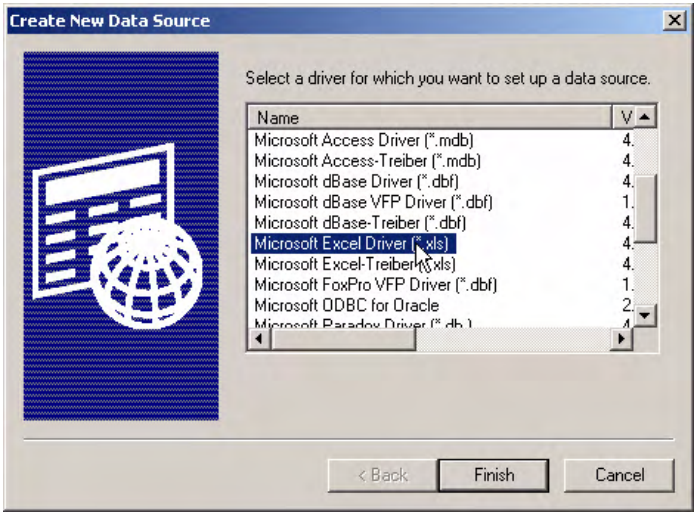
1. From the **System DSN** tab select **Add** to create a new ODBC link:

Figure E-19 Add the Excel ODBC Driver



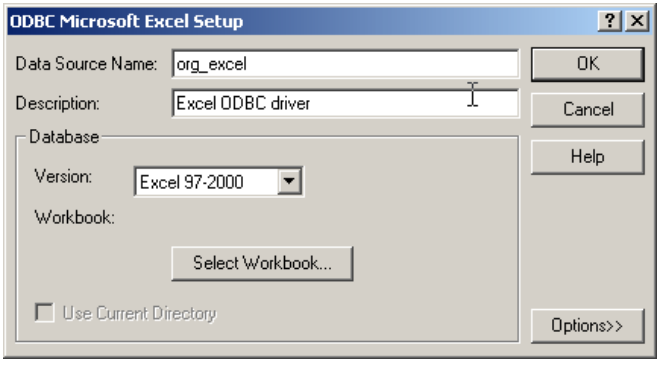
2. Select the **Microsoft Excel Driver**. It has an *xls extension.

Figure E-20 New Data Source



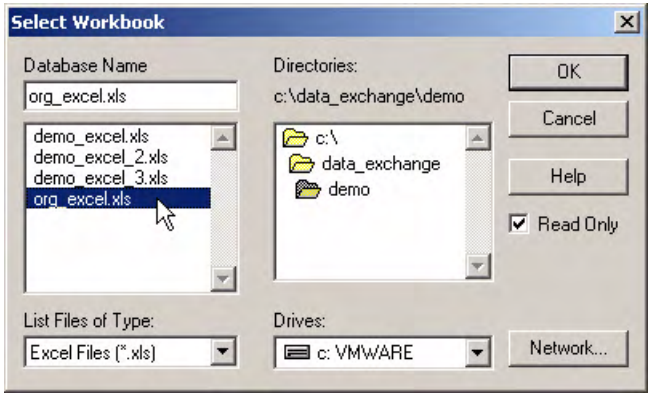
3. Click **Finish**. The Microsoft Excel Setup dialog box appears:

Figure E-21 Excel ODBC_DSN



4. Type the **Data Source Name** and a **Description**. The Data Source Name must match the **DSN** name in your extractor configuration file. The description is for your own reference.
5. Click **Select Workbook** to specify the data source that you intend to use. The Select Workbook dialog box appears:

Figure E-22 Selecting the Workbook



6. Locate the workbook, then click **OK** to return to the Microsoft Excel Setup dialog box.
7. Click **OK** to finish the procedure.

Configuring the Extractor and the Import Mapping

This section describes the example org_excel.ini extractor configuration file and the org_excel import mapping, giving special attention to the way entity relations are handled.

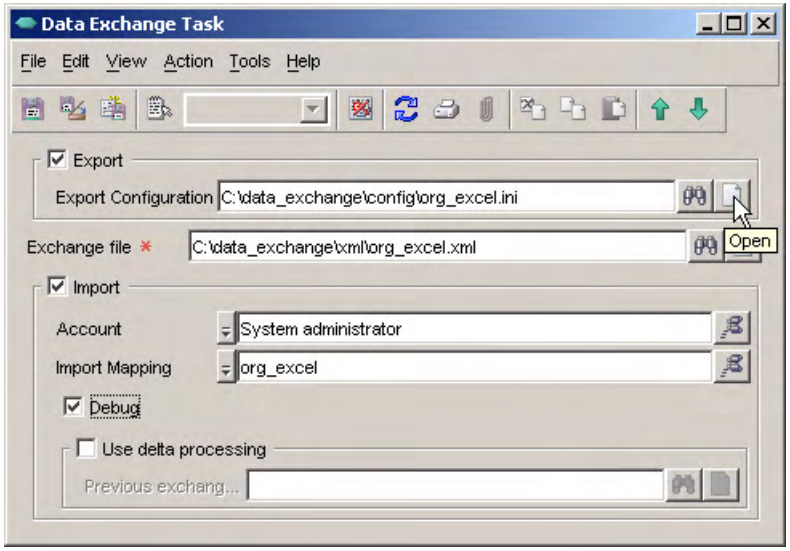
This example imports N:N relations using the approach that provides optimum performance. Please note that when importing large quantities of data, your system may have insufficient memory to complete the import process. If so, you should use the alternative approach described in “Importing N:N Relations as 1:Rel:1 Relations” on page 195.

Mapping data from the XML file to objects and attributes in the OpenView database can be a labor-intensive job. If you cannot use one of the import mapping examples provided, you can make the process easier by creating a well-structured extractor configuration file and making sure it works before you start the import mapping process. If you don’t do this, you may end up having to create the import mapping more than once. To better demonstrate how what is entered in the configuration file affects what needs to be entered in the import mapping, this example presents a description of the import mapping for each class immediately after an explanation of the statements that define the class in the extractor configuration file. In reality, you usually create the entire configuration file and then define the import mapping based on the information in the XML file.

The export process can be started from the command line or from the OpenView console (see “Create a Data Exchange Task” on page 42 for additional information). You can open the configuration file from the

Data Exchange Task dialog box. Click **Open** to edit or view the extractor configuration file selected in the **Export Configuration** field as shown in Figure E-23:

Figure E-23 Org_Excel Data Exchange Task



In this example, the following classes are configured for export by the extractor:

Class	Description
ORGANIZATION	This class exports the names and search codes of organizations from the ORG table in the data source.
PERSON	This class exports the search codes, first names, last names, and organizations of employees from the EMPLOYEE table in the data source.
PHONE	This class exports the employee telephone numbers from the EMPLOYEE table in the data source.

Class	Description
EQUIPMENT	This class exports the different hardware items from the EQUIPMENT table in the data source.
OWNERS	This class exports the relations between hardware owners and equipment from the EQUIPMENT table in the data source.

Example E-1 shows the org_excel extractor configuration file used in this example:

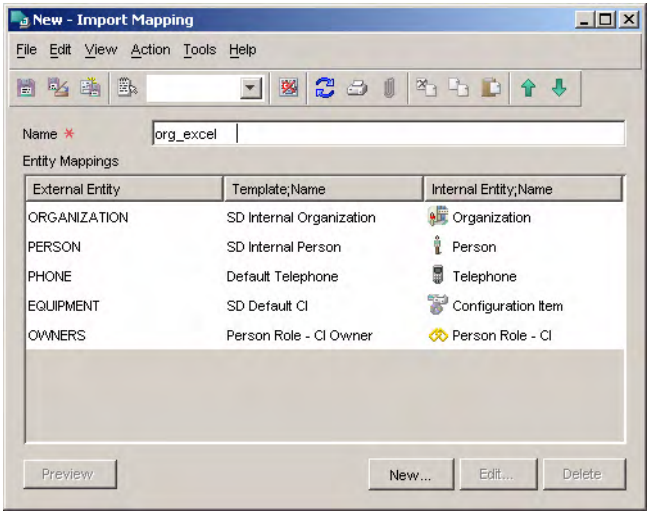
Example E-1 N:N Relations Sample Configuration File

```
[DSN]
NAME=org_excel
USR=
PWD=

[SYSTEM]
LOG=TRUE
XML=TRUE
LOG_FILE=C:\data_exchange\log\org_excel.log
XML_OUTPUT_FILE=C:\data_exchange\xml\org_excel.xml
APPLICATION_NAME=Microsoft Excel

[CLASSES]
NAME=ORGANIZATION, PERSON, PHONE, EQUIPMENT, OWNERS

[ORGANIZATION]
SOURCE=[ORG]
ATT=[SEARCHCODE], [NAME]
COLUMNS=DISTINCT [SEARCHCODE], [NAME]
LOADTABLE=TRUE

[PERSON]
SOURCE=[EMPLOYEE]
ATT=[FIRSTNAME], [LASTNAME], [ORG], [SEARCHCODE2]
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [ORG], [EMPLOYEE].[SEARCHCODE] AS
[SEARCHCODE2]
LOADTABLE=TRUE
```

```
[PHONE]
SOURCE=[EMPLOYEE]
ATT=[EMP_SEARCHCODE], [NUMBER]
COLUMNS=DISTINCT [EMPLOYEE].[SEARCHCODE] AS [EMP_SEARCHCODE], [NUMBER]
LOADTABLE=TRUE

[EQUIPMENT]
SOURCE=[EQUIPMENT]
ATT=[NAME], [USER], [SEARCHCODE3], [TYPE]
COLUMNS=DISTINCT [NAME], [USER], [EQUIPMENT].[SEARCHCODE] AS [SEARCHCODE3],
[TYPE]
LOADTABLE=TRUE
PARENT=PERSON
PARENT_RELATION=[USER]=[SEARCHCODE2]
PARENT_RELATION_NAME=USER

[OWNERS]
SOURCE=[EQUIPMENT]
ATT=[EQUIP_SEARCHCODE], [OWNER]
COLUMNS=DISTINCT [EQUIPMENT].[SEARCHCODE] AS [EQUIP_SEARCHCODE], [OWNER]
LOADTABLE=TRUE
CONDITION=[OWNER] IS NOT NULL
```

Figure E-24 shows the import mapping used in this example. The mapping contains a list of entity mappings, one for each class defined in the extractor configuration file:

Figure E-24 Import Mapping



Configuring the DSN Section

The NAME line in the DSN section of the configuration file is the Data Source Name you entered when you created the ODBC link. Each configuration file needs to have its own ODBC link with a unique Data Source Name.

Configuring the SYSTEM Section

When you start data exchange tasks from the user interface, you only need to fill in the LOG, XML, and APPLICATION NAME fields. You only need to supply values for LOG_FILE, OUTPUT_FILE, and XML_OUTPUT_FILE if you execute data exchange tasks from a command line.

Configuring the CLASSES Section

When developing the configuration file, the order that the classes are listed in the CLASSES section is important. The sections that follow should mirror the order in the CLASSES section. The extractor takes each class defined in the CLASSES section and makes SQL statements. Sources, columns and classes are selected and relationships made according to the definitions in the configuration file (see “SQL Statements in the Data Extraction Process” on page 32 for additional information). When you later import the data, the objects are imported in the same order in which they they were exported. The order requires some thought when developing the configuration file for importing relations. For example, the configuration file in Example E-1 extracts the ORGANISATION class before the PERSON class because the ORG attribute of the PERSON class is an entity reference to the ORGANISATION class.

NOTE

If a class is not specified on the NAMES line in the CLASSES section of the configuration file, it is not imported, even if the class is described lower down in the configuration file. No error message is displayed if a class is missing. A missing class can only be detected by a manual check of the configuration file, or seeing that it was not exported to the XML file or imported into the OpenView database.

Configuring the ORGANIZATION Class

The ORGANISATION class is based on the ORG table in the data source. This shows that you can give a class any name you want in the configuration file as long as the SOURCE is correctly identified. Keep in mind that when the exported data is put into the XML file, it is sorted under the class name that you give it, and that class name is what you need to map to an object in the OpenView database.

The ORGANISATION class is intentionally put in the configuration file before the PERSON class because PERSON exports the ORG external attribute, which is a reference to the NAME external attribute of the ORGANIZATION class. By exporting ORGANISATION before PERSON, you ensure that the organizations already exist in the database when the employees are imported. This ensures that each employee is associated with the correct organization. This is an example of how to handle N:1 relations. For further information about N:1 relations, see See “N:1 Relations Based on a Search Key” on page 170.

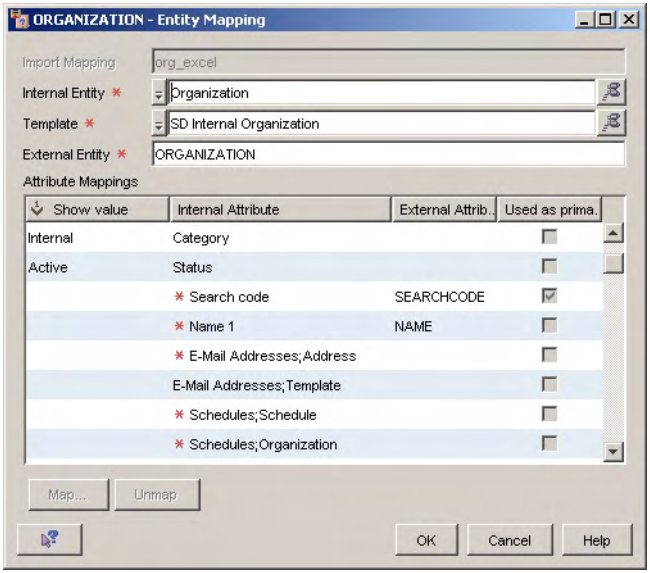
Import Mapping for the ORGANIZATION Class

In the import mapping, the ORGANISATION class is mapped to the Organization object type in the OpenView database. The SD Default Organization template is used for the mapping. The SEARCHCODE2 external attribute is mapped to the Search code attribute and the NAME external attribute is mapped to the Name1 attribute. Search code is used as the unique key for identifying the class.

The template specifies default values for certain attributes that are not defined in the data source. For example, the template assigns the value Active to the Status attribute, and the value Internal to the Category attribute.

Figure E-25 displays the complete import mapping:

Figure E-25 Import Mapping - ORGANIZATION



Configuring the PERSON Class

The PERSON class is configured to be imported before the PHONE class. This is because phone numbers contain a reference to employees and so the employee information must be imported first. The PERSON class is also configured to be imported before the EQUIPMENT class. This is because the USER attribute of the EQUIPMENT class has an N:N (parent-child) relation to the PERSON class, and the parent PERSON class must be exported before the child EQUIPMENT class. For more information about N:N relations, see “N:N (Parent-Child) Relations” on page 171.

The SEARCHCODE external attribute is exported with the alias SEARCHCODE2 to avoid confusion. All three tables in the database source have external attributes called SEARCHCODE, and aliasing is used to export each one with a unique attribute name - SEARCHCODE, SERCHCODE2, and SEARCHCODE3. The exported attribute names do not need to be unique, because they belong to distinct classes. However, exporting them with unique names helps to avoid confusion, especially when defining the import mappings.

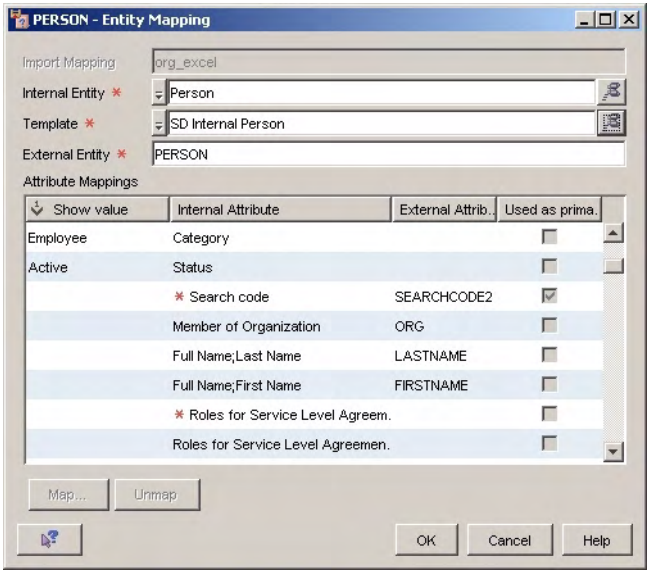
Import Mapping for the PERSON Class

In the import mapping, the PERSON class is mapped to the Person object type in the OpenView database. The SD Internal Person template is used for this class. The SEARCHCODE2 external attribute is mapped to the Search code attribute, ORG is mapped to the Member of Organization attribute, LASTNAME is mapped to Full name;Last Name, and FIRSTNAME is mapped to Full name;First Name. Search code is used as the unique key for identifying the class.

The template specifies default values for certain attributes that are not defined in the data source. For example, the template assigns the value Active to the Status attribute, and the value Employee to the Category attribute.

Figure E-26 displays the complete import mapping:

Figure E-26 Import Mapping - PERSON



You need to create an entity reference for the ORG external attribute. The values of the ORG attribute correspond to the values of the NAME attribute in the ORGANIZATION class, which is mapped to the Name1 attribute of the Organization object type in the OpenView database (see Figure E-25). Therefore, the entity reference you create must map the ORG external attribute to the Name1 attribute of the Organization object type in the OpenView database.

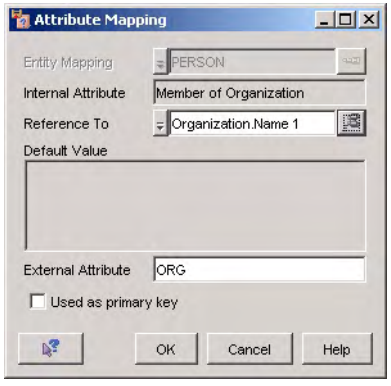
To create the entity reference:

1. In the Entity Mapping dialog box for the PERSON class, double-click the **Member of Organization** field.

   The Attribute Mapping dialog box appears.

2. Type ORG in the **External Attribute** field.
3. Click the **Quick Find** button next to the **Reference To** field, and then choose the Organization:Name1 search code.
4. Click **OK** to finish.

Figure E-27 Attribute Mapping - ORG



Configuring the PHONE Class

Information in the PHONE class is exported from the EMPLOYEE table in the data source and imported to the Telephone object type in the OpenView database. The configuration file shown in Example E-1 shows that the ATT name for the employee search code (EMP_SEARCHCODE) does not match its COLUMN name in the data source (SEARCHCODE). The ATT attribute names are decided by you, just like class names. If you want to export a COLUMN name in the data source to a different attribute name in the XML file, you need to specify an alias using the AS statement in the COLUMN line. In this example, the ATT name EMP_SEARCHCODE is exported from the COLUMN called SEARCHCODE and listed in the XML file as EMP_SEARCHCODE.

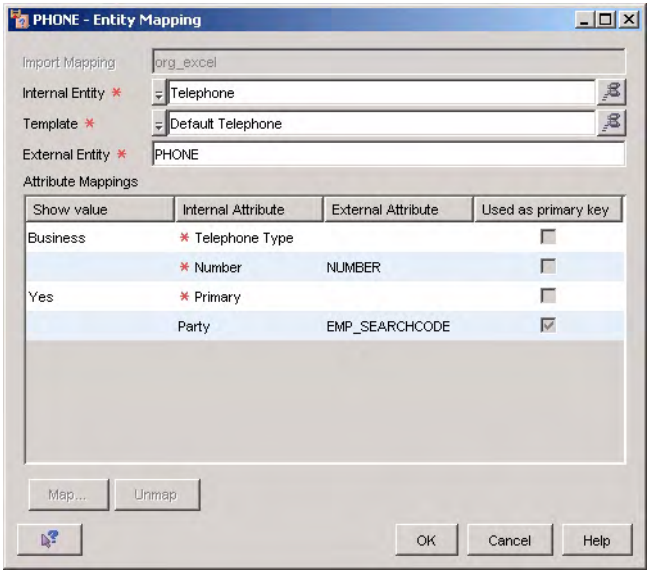
Import Mapping for the PHONE Class

In the import mapping, the PHONE class is mapped to the Telephone object type in the OpenView database. The Default Telephone template is used for this class. The EMP_SEARCHCODE external attribute is mapped to the Party attribute, and the NUMBER external attribute is mapped to the Number attribute. EMP_SEARCHCODE is used as a unique key.

The template specifies default values for certain attributes that are not defined in the data source. For example, the template assigns the value Business to the Telephone Type attribute, and the value Yes to the Primary attribute.

Figure E-28 displays the complete import mapping:

Figure E-28 Import Mapping - PHONE



You need to create an entity reference for the EMP_SEARCHCODE attribute. The values of the EMP_SEARCHCODE attribute correspond to the values of the SEARCHCODE2 external attribute in the PERSON class, which is mapped to the Search Code attribute of the Person object type in the OpenView database (see Figure E-26). Therefore, the entity reference you create must map the EMP_SEARCHCODE external attribute to the Party:Searchcode attribute of the Person object type in the OpenView database.

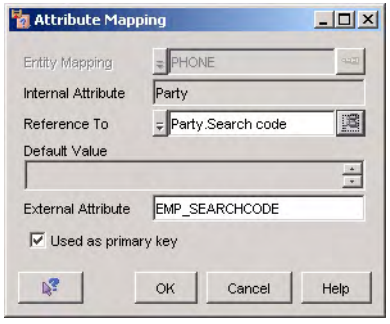
To create the entity reference:

1. In the Entity Mapping dialog box for the PHONE class, double-click the **Party** field.

   The Attribute Mapping dialog box appears.

2. Type `EMP_SEARCHCODE` in the **External Attribute** field.
3. Click the **Quick Find** button next to the **Reference To** field, and then choose Party:Searchcode.
4. Select the **Used as primary key** check box.
5. Click **OK** to finish.

Figure E-29 Attribute Mapping - EMP_SEARCHCODE



Configuring the EQUIPMENT Class

The EQUIPMENT class is based on the EQUIPMENT table in the data source. This is the child class of an N:N (parent-child) relation, and is listed for extraction after the PERSON class, which is its parent. The EQUIPMENT class is intentionally listed for extraction before the OWNERS class, which contains references to attributes in the EQUIPMENT class.

The PARENT_RELATION statement associates the USER external attribute of the child class with the SEARCHCODE2 external attribute of the parent class. PARENT_RELATION_NAME must identify the external attribute of the child class. In this example, it is USER.

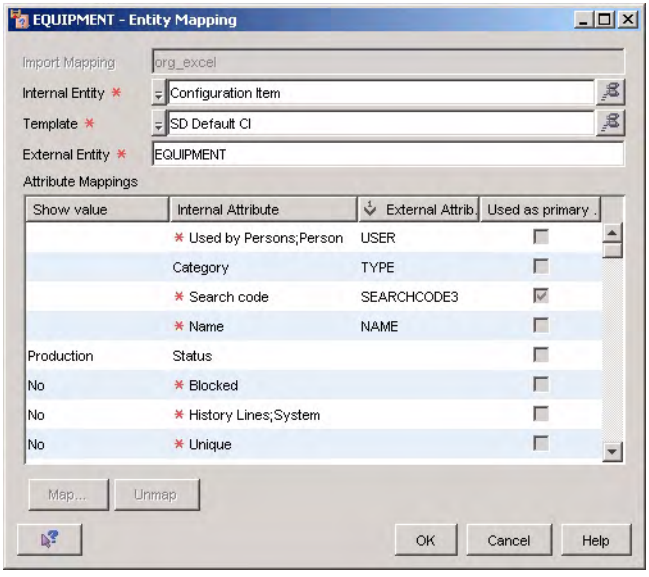
Import Mapping for the EQUIPMENT Class

In the import mapping, the EQUIPMENT class is mapped to the Configuration Item object type in the OpenView database. The SD Default CI template is used for this class. The SEARCHCODE3 external attribute is mapped to the Search code attribute, TYPE is mapped to the Category attribute, NAME is mapped to the Name attribute, and USER is mapped to its parent Used by Persons;Person attribute.

The template specifies default values for certain attributes that are not defined in the data source. For example, the template assigns the value Production to the Status attribute.

Figure E-30 displays the complete import mapping:

Figure E-30 Import Mapping - EQUIPMENT



You need to create an entity reference for the USER external attribute.

To create the entity reference:

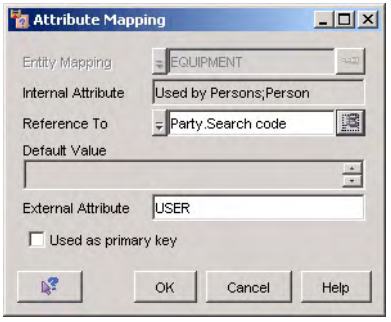
1. In the Entity Mapping dialog box, double-click the **Used by Persons;Person** field.

   The Attribute Mapping dialog box appears.

2. Type USER in the **External Attribute** field.

3. Click the Quick Find button next to the **Reference To** field and choose the **Party:Searchcode** attribute.
4. Click **OK** to finish.

Figure E-31 Attribute Mapping - USER



You need to create value mappings for the TYPE external attribute.

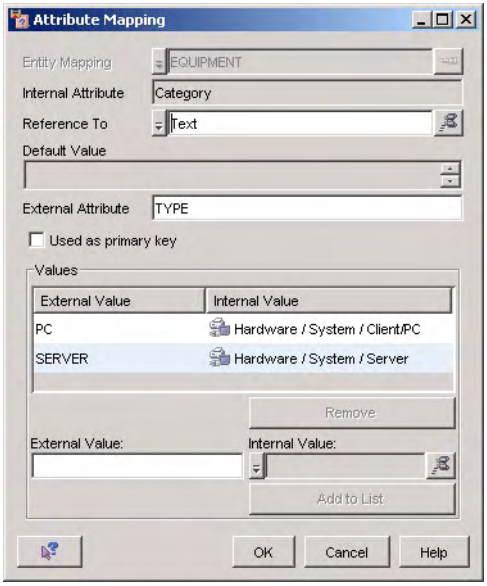
To create the value mapping:

1. In the Entity Mapping dialog box for the EQUIPMENT class, double-click the **Category** field.

   The Attribute Mapping dialog box appears.
2. Type `TYPE` in the **External Attribute** field.
3. Add a list of value mappings. For example, to provide a value mapping for the PCs, type PC in the **External Value** field, choose the required CI category in the **Internal Value** field, then click **Add to List**. Repeat the process for the servers.

Figure E-32 shows the category value mappings created in this example:

Figure E-32 Value Mapping - TYPE



4. Click **OK** to finish.

Configuring the OWNERS Class

The OWNERS class defines the relations between the PCs and persons designated as owners. This is a 1:Rel:1 relation. The OWNERS class is mapped to the Person Role - CI object type that contains a number of different attributes as fields. It enables you to specify the nature of the relation of the classes that are linked to this class. The CONDITION statement ensures that information is not exported for configuration items without a registered owner (such as ORGPC007).

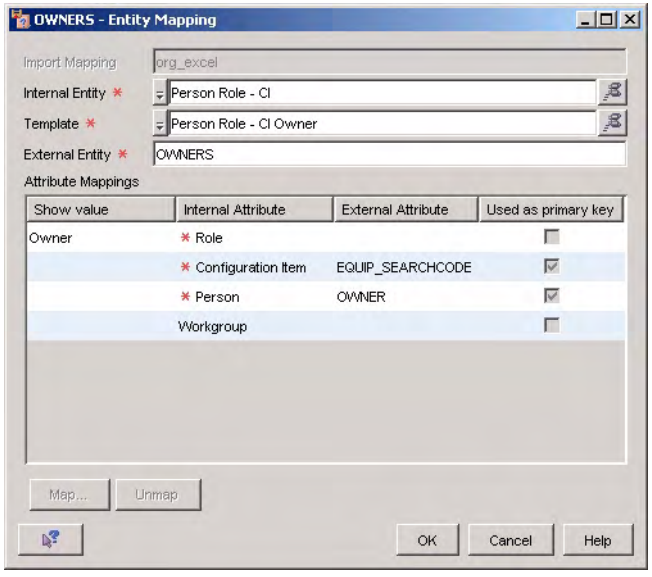
Import Mapping for the OWNERS Class

In the import mapping, the OWNERS class is mapped to the Person Role - CI object type in the OpenView database. The Default Person Role - CI template is used for this class. The EQUIP_SEARCHCODE external attribute is mapped to the Configuration Item attribute, and OWNER is mapped to the Person attribute.

The template specifies default values for certain attributes that are not defined in the data source. For example, the template assigns the value Owner to the Role attribute.

The completed import mapping is displayed in the following dialog box:

Figure E-33 Import Mapping - OWNERS



NOTE The Role attribute is mandatory and should be specified either as a default value in the template used, or by the imported class.

You need to create entity references for the attribute mappings for EQUIP_SEARCHCODE and OWNER. In common with many relation object types, the Person Role - CI object type has no single unique field (such as a search code). Instead, each pairing that relates two other object types (in this case, a configuration item and a person) is unique. Therefore, both those attribute mappings must be defined as a primary key, as described below.

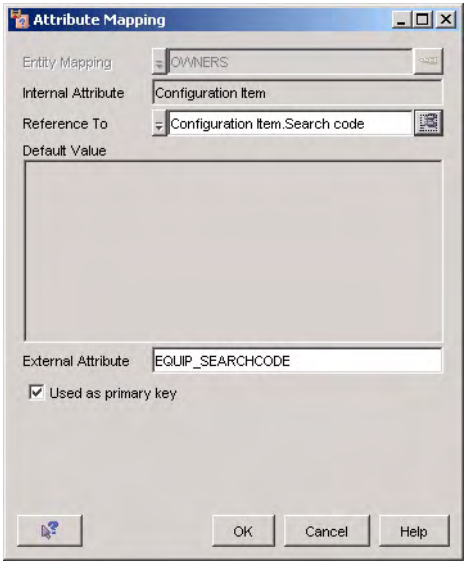
To create the entity references:

1. In the Entity Mapping dialog box, double-click the **Configuration Item** field.

   The Attribute Mapping dialog box appears.

2. Type `EQUIP_SEARCHCODE` in the **External Attribute** field.
3. Click the Quick Find button next to the **Reference To** field and choose the **ConfigurationItem:Searchcode** attribute.
4. Select the **Used as Primary Key** check box.

Figure E-34 Attribute Mapping - EQUIP_SEARCHCODE



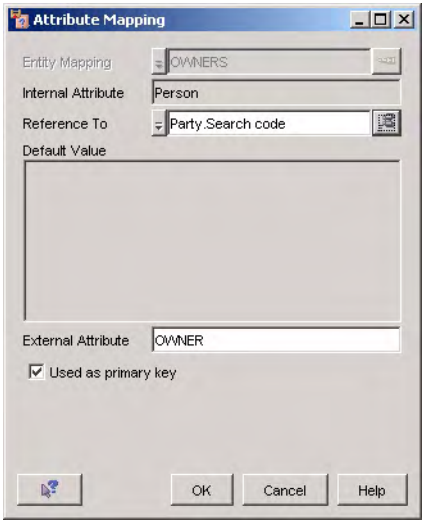
5. Click **OK** to return to the Entity Mapping dialog box.
6. In the Entity Mapping dialog box, double-click the **Person** field.

   The Attribute Mapping dialog box appears.

7. Type `OWNER` in the **External Attribute** field.
8. Click the Quick Find button next to the **Reference To** field and choose the **Party:Searchcode** attribute.

9. Select the **Used as Primary Key** check box.

Figure E-35 Attribute Mapping - OWNER



10. Click **OK** to finish.

Importing the Data

Do not import the data until after you have taken the time to verify that everything was exported correctly. You can run the task to import data from the OpenView console or from a command line, see “Create a Data Exchange Task” on page 42.

During the data import process, workflow functionality takes place in the same was as for information created manually through a client connection. In this example, if a person imported as a CI owner is a member of an organization, workflow functionality automatically registers the organization as the CI owner organization.

Importing N:N Relations as 1:Rel:1 Relations

This section explains the changes you need to make to the org_excel.ini extractor configuration file and the org_excel import mapping if you want to handle the N:N relation for configuration item users as though they were 1:Rel:1 relations. You may need to use this approach when

importing large quantities of data on a system with insufficient memory to handle the approach explained in “Configuring the Extractor and the Import Mapping” on page 179.

The following table lists the changes to the way classes are configured for export by the extractor:

Class	Description
ORGANIZATION	No change
PERSON	No change.
PHONE	No change.
EQUIPMENT	This class exports the different hardware items from the EQUIPMENT table in the data source. The USER external attribute is not exported in this class. Information about hardware users is exported to the USERS class instead.
USERS	This class is added for the purpose of exporting the relations between hardware users and equipment from the EQUIPMENT table in the data source.
OWNERS	No change.

Example E-2 shows the modified org_excel extractor configuration file. Notice that the EQUIPMENT class does not contain PARENT, PARENT_RELATION, or PARENT_RELATION_NAME statements. Instead, an additional class (USERS) is defined. Notice also the similarity between the USERS class and the OWNERS class:

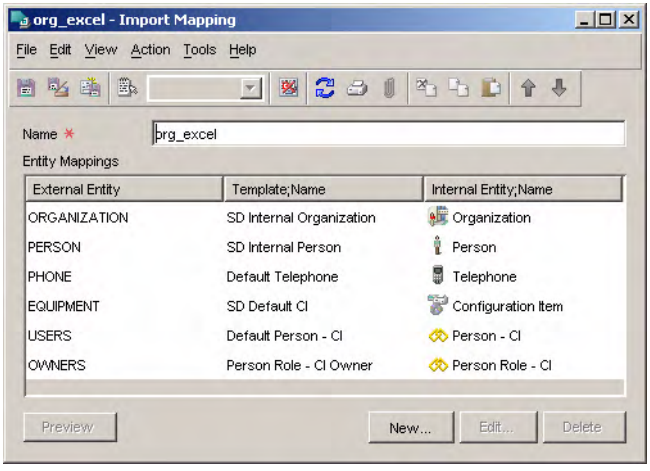
Example E-2 1:Rel:1 Relations Sample Configuration File

```
[DSN]
NAME=org_excel
USR=
PWD=

[SYSTEM]
LOG=TRUE
XML=TRUE
```

```
LOG_FILE=C:\data_exchange\log\org_excel.log
XML_OUTPUT_FILE=C:\data_exchange\xml\org_excel.xml
APPLICATION_NAME=Microsoft Excel

[CLASSES]
NAME=ORGANIZATION, PERSON, PHONE, EQUIPMENT, USERS, OWNERS

[ORGANIZATION]
SOURCE=[ORG]
ATT=[SEARCHCODE], [NAME]
COLUMNS=DISTINCT [SEARCHCODE], [NAME]
LOADTABLE=TRUE

[PERSON]
SOURCE=[EMPLOYEE]
ATT=[FIRSTNAME], [LASTNAME], [ORG], [SEARCHCODE2]
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [ORG], [EMPLOYEE].[SEARCHCODE] AS
[SEARCHCODE2]
LOADTABLE=TRUE

[PHONE]
SOURCE=[EMPLOYEE]
ATT=[EMP_SEARCHCODE], [NUMBER]
COLUMNS=DISTINCT [EMPLOYEE].[SEARCHCODE] AS [EMP_SEARCHCODE], [NUMBER]
LOADTABLE=TRUE

[EQUIPMENT]
SOURCE=[EQUIPMENT]
ATT=[NAME], [SEARCHCODE3], [TYPE]
COLUMNS=DISTINCT [NAME], [EQUIPMENT].[SEARCHCODE] AS [SEARCHCODE3], [TYPE]
LOADTABLE=TRUE

[USERS]
SOURCE=[EQUIPMENT]
ATT=[EQUIP_SEARCHCODE], [USER]
COLUMNS=DISTINCT [EQUIPMENT].[SEARCHCODE] AS [EQUIP_SEARCHCODE], [USER]
LOADTABLE=TRUE

[OWNERS]
SOURCE=[EQUIPMENT]
ATT=[EQUIP_SEARCHCODE], [OWNER]
COLUMNS=DISTINCT [EQUIPMENT].[SEARCHCODE] AS [EQUIP_SEARCHCODE], [OWNER]
LOADTABLE=TRUE
CONDITION=[OWNER] IS NOT NULL
```

Figure E-36 shows the modified import mapping used in this example. The mapping contains a list of entity mappings, one for each class defined in the modified extractor configuration file:

Figure E-36 Modified Import Mapping



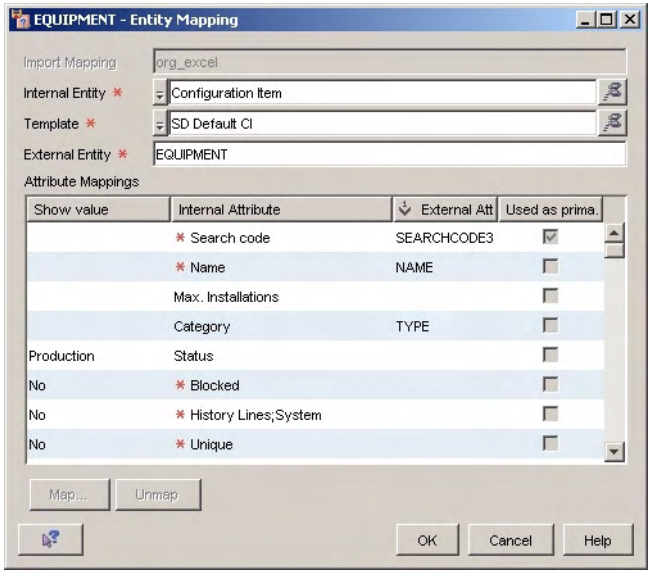
Configuring the EQUIPMENT Class

The EQUIPMENT class is based on the EQUIPMENT table in the data source. The EQUIPMENT class is intentionally listed for extraction before the USERS and OWNERS classes, which contain references to attributes in the EQUIPMENT class.

Import Mapping for the EQUIPMENT Class

Because information about configuration item users is exported to a separate USERS class, the import mapping for the EQUIPMENT class does not include a mapping for the USER external attribute. Figure E-30 displays the modified import mapping:

Figure E-37 Import Mapping - EQUIPMENT



Configuring the USERS Class

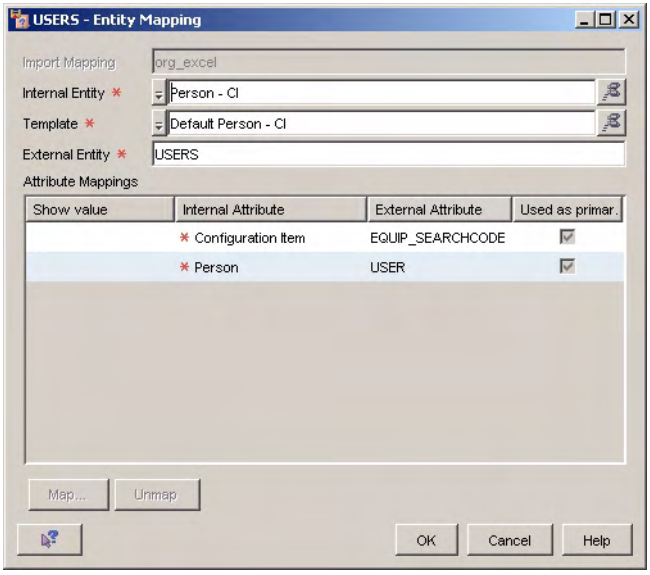
The USERS class defines the relations between equipment and users. It is created separately so that it can be imported to a different object type (Person - CI) in the OpenView database. The EQUIPMENT class is imported as configuration item objects. The USERS class is imported as Person - CI objects in the OpenView database.

Import Mapping for the USERS Class

In the import mapping, the USERS class is mapped to the Person - CI object type in the OpenView database. The Default Person - CI template is used for this class. The EQUIP_SEARCHCODE external attribute is

mapped to the Configuration Item attribute. The USER external attribute is mapped to the Person attribute. Figure E-38 displays the complete import mapping:

Figure E-38 Import Mapping - USERS



You need to create entity references for the EQUIP_SEARCHCODE and USER external attributes.

To create the entity references:

1. In the Entity Mapping dialog box for the USERS class, Double-click the **Configuration Item** field.

   The Attribute Mapping dialog box appears.

2. Type `EQUIP_SEARCHCODE` in the **External Attribute** field.

3. Click the **Quick Find** button next to the **Reference To** field, and choose the **ConfigurationItem:Searchcode** attribute.

4. Select the **Used as Primary Key** check box.

5. Click **OK** to return to the Entity Mapping dialog box.

6. Double-click the **Person** field.

   The Attribute Mapping dialog box appears.

7. Type `USER` in the **External Attribute** field.
8. Click the **Quick Find** button next to the **Reference To** field, and choose the **Party:Searchcode** attribute.
9. Select the **Used as Primary Key** check box.
10. Click **OK** to finish.

Importing Data From an ASCII Text File

Microsoft's ODBC Text driver can be used to export data from an ASCII file into an XML file for importing. This can be a useful way of importing data that is not stored in an ordinary SQL database; Oracle, SQL Server or Microsoft Access, for example.

Step 1. Configure the `SCHEMA.INI` file for the ODBC driver.

1. Create a new ASCII text file in a text editor and insert the following text:

```
FirstName,LastName,JobTitle,Searchode
Irvine,Welsh,Manager,IWELSH
Jonathan,Cape,Specialist,JCAPE
```

   Save the file under the name `sample.txt`.

2. Create a `SCHEMA.INI` file with the following contents:

```
[sample.txt]
ColNameHeader=True
Format=CSVDelimited
MaxScanRows=1
CharacterSet=OEM
Col1=FIRSTNAME Char Width 255
Col2=LASTNAME Char Width 255
Col3=JOBTITLE Char Width 255
Col4=SEARCHCODE Char Width 255
```

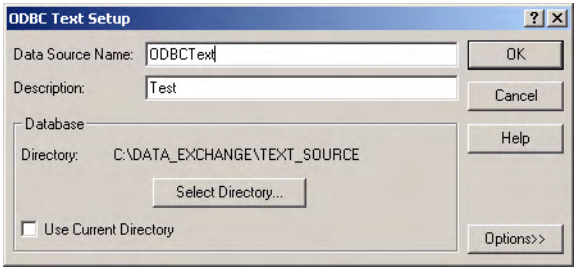
   The `SCHEMA.INI` file is used to define the ASCII text format. Place the file in the same directory as the `sample.txt` file.

NOTE The Help files for the Microsoft Text driver contain detailed information on how to define your text file with the SCHEMA.INI file. From the Microsoft driver help, the topics "Defining Text Format" and "Schema.ini file" are particularly useful.

Step 2. Setup the ODBC Text driver.

1. Start ODBC Data Sources Administrator from the Windows Control Panel.
2. Select the **System DSN** tab and click **Add**.
3. Select the **Microsoft Text Driver**, then click **Finish**.
4. Enter the **Data Source Name**.
5. Enter a **Description** of the file.
6. Clear the **Use Current Directory** check box and click **Select Directory**. The directory selected needs to be the same directory in which the `SCHEMA.ini` file and the `sample.txt` file are located. The ODBC Text Setup dialog box should be set up similar to the following dialog box:

Figure E-39 ODBC Text Setup



Step 3. Modify the extractor configuration file.

1. Copy the sample extractor configuration file `sd_event.ini` in Service Desk and save it with a new name. For example, `ODBCText.ini`.
2. Modify the extractor configuration file as follows:

```
[DSN]
NAME=ODBCText
USR=
PWD=

[SYSTEM]
LOG=TRUE
XML=TRUE
TXT=FALSE
LOG_FILE=C:\data_exchange\log\ODBCText.log
```

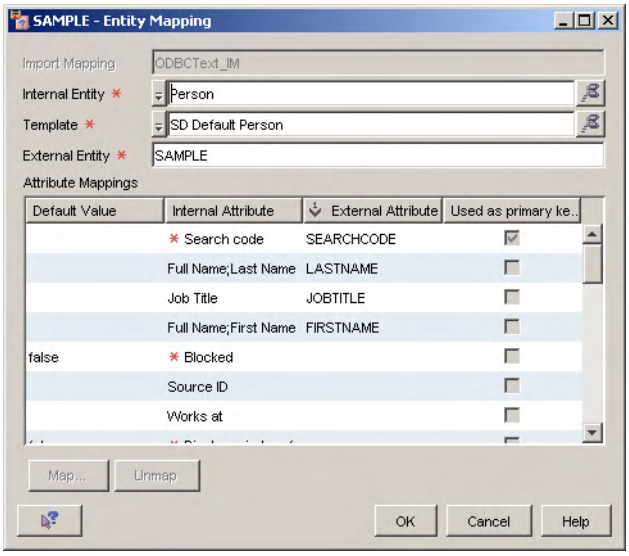
```
XML_OUTPUT_FILE=C:\data_exchange\xml\ODBCText.xml
APPLICATION_NAME=ODBCText
ENCODING=ISO-8859-1

[CLASSES]
NAME=SAMPLE

[SAMPLE]
SOURCE=[sample.txt]
ATT=[FIRSTNAME], [LASTNAME], [JOBTITLE], [SEARCHCODE]
COLUMNS=DISTINCT [FIRSTNAME], [LASTNAME], [JOBTITLE], [SEARCHCODE]
LOADTABLE=TRUE
```

Step 4. Export the ASCII text file data.

1. In the **OV Configuration** workspace group, expand the **Data** →**Data Exchange**→**Data Exchange Task** branch. Right-click in the Data Exchange Task window and select **New Data Exchange Task**.
2. Select the **Export** check box.
3. In the **Export Configuration** field, type the name of the extractor configuration file you modified. In this example it is `ODBCtext.ini`.
4. In the **Exchange file** field, type the name of the XML file you specified in the `ODBCtext.ini` file. In this example it is `ODBCText.xml`.
5. Click **File**→**Save** to save this task for use another time.
6. Click **Action**→**Start** to export the data from the text file.

Step 5. Create an import mapping for the data. In this example, the mapping is called `ODBCText_IM`. Figure E-40 shows the import mapping for the SAMPLE class being imported in this example:

Figure E-40 ODBC Text file - Import Mapping



The SAMPLE class is mapped to the Person object type in the OpenView database. SD Default Template is the name of the template used for the creation of the new Person objects.

Step 6. Import the data.

1. In the **OV Configuration** workspace group, expand the **Data →Data Exchange→Data Exchange Task** branch. Right-click in the Data Exchange Task window and select **New Data Exchange Task**.
2. In the **Exchange file** field, enter the location and name of the `ODBCText.xml` file.
3. Select the **Import** check box.
4. Enter an account created for integration purposes.
5. In the **Import mapping** field, use the **Quick Find** button to find the import mapping you created for importing the text file. In this example it is `ODBCText_IM`.

6. Select the **Debug** check box and click **Action→Start** to import the data.
7. Check `ODBCText_IM_imp.log` to verify that the import was successful.


invent


Java™
COMPATIBLE