

# **HP OpenView Service Desk 3.0**

## **Data Exchange Administrator's Guide**

**Second Edition, First Revision**



**Manufacturing Part Number: N/A**

**December 2000**

---

## Legal Notices

*Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.*

**Restricted Rights Legend.** Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company  
3000 Hanover Street  
Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c)(1,2).

**Copyright Notice.** © Copyright 2000 Hewlett-Packard Company

### Trademark Notices

Java™ is a U.S. trademark of Sun Microsystems, Inc.

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

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

SQL\*Net® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

SQL\*Plus® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

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

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

**Preface****1. The Data Exchange Concept**

The Big Picture .....	27
Overview of the Export Process.....	30
Configuring the Import Mapping .....	31
Importing the Data .....	32

**2. Exporting Data**

Deciding What to Export.....	35
Checking the Database Structure .....	35
Configuring the Extractor.....	37
Modifying the Configuration File.....	37
Configuration File Keywords .....	39
Defining Relations in the Configuration File .....	41
Search Key; N:1 Relations .....	41
Parent-Child; N:N Relations.....	43
Relations with Relation Type; 1:Rel:1 Relations.....	44
Defining the ODBC link .....	46
Special Instructions for Network Node Manager .....	47
The Extraction Process .....	49
Exporting Data.....	51
Using a Task to Export Data From a Storage Device.....	51
Creating a Task to Export Data.....	53
Exporting From the Command Line .....	53
The Data Exchange Files .....	53
The XML Format.....	54
Viewing Data Exchange Files .....	57

**3. The Import Mapping**

About Item Mapping .....	62
Attribute Mapping .....	62
Key Binding.....	62
Value Mapping.....	63
Defining Relations .....	63
Search Key; N:1 Relations .....	64
Parent-Child; N:N Relations.....	67
Relations with Relation Type; 1:Rel:1 Relations.....	70

---

# Contents

Mapping Service Events .....	75
Mapping Classes and Attributes .....	77
Creating a New Import Mapping .....	77
Modifying an Import Mapping .....	78

## 4. Importing Data in Batches

Using a Task to Import Data .....	87
Creating Data Exchange Tasks .....	90
Executing a Task .....	90
Scheduling a Data Exchange Task .....	91
Command Line Parameters .....	92
Parameters for Exporting .....	92
Parameters for Importing .....	93
Parameters for Importing and Exporting .....	93
Integration Accounts .....	95

## 5. Importing Service Events

The Service Event Import Process .....	100
Service Desk Event Command Line .....	101
Special Characters .....	102
Using the Service Event Configuration File .....	102
Service Event Examples .....	104
Resending Service Events .....	105
Preventing Event Storms .....	107
Queuing Events .....	107
Creating a Queue .....	107
Enqueue .....	108
Dequeue .....	108
Queuctl .....	108
Addentry .....	109
Service Desk Event Communicator and Rule Manager .....	110

## 6. Outbound Service Events

The Outbound Service Event Process .....	112
The Rule Manager User Interface .....	114
Creating a Database Rule to Send a Service Event .....	114

Resending Failed Service Events . . . . .	122
<b>7. Auditing and Troubleshooting</b>	
The Viewer, Log Files, and the Debug Mode . . . . .	124
Troubleshooting . . . . .	125
<b>A. Integration With Network Node Manager</b>	
Integration Possibilities . . . . .	132
Creating Incidents in Service Desk . . . . .	132
Importing NNM Nodes into Service Desk . . . . .	132
Installation . . . . .	133
Service Desk Application Server . . . . .	133
Configuration . . . . .	134
Configuring Network Node Manager . . . . .	134
Select an NNM Event . . . . .	134
Modify an Event . . . . .	134
Configuring Service Desk . . . . .	136
Configure the Sd_event File . . . . .	137
Modify the Import Mapping . . . . .	137
NNM Variables . . . . .	137
Special Characters . . . . .	138
Sequential Attribute Variables . . . . .	138
Special Information Variables . . . . .	139
SNMP-Specific Variables . . . . .	140
Example of NNM on UNIX Server . . . . .	142
Example of NNM on UNIX . . . . .	142
Managing Event Storms on UNIX . . . . .	142
<b>B. Integration With ITO</b>	
Integration Possibilities . . . . .	146
Automatic Forwarding of Messages . . . . .	147
Automatic Forwarding of Acknowledgment Messages . . . . .	147
Automatic Forwarding of Message Annotations . . . . .	147
Manually Forwarding Messages . . . . .	148
Installation . . . . .	149
Service Desk . . . . .	149
Integrations . . . . .	149
Demo Database . . . . .	150

---

# Contents

ITO.....	150
ITO Server Account.....	150
The HPOVSD Depot.....	151
Multiple Servers.....	154
Configuration.....	155
Make Service Desk an ITO User.....	155
Create Configuration Items.....	156
Import Mapping.....	158
Attribute Mapping.....	158
The Event Queue.....	160
Configure the Ini File.....	160
Configure the Trouble Ticket Interface.....	161
Database Rules.....	161
Creating New Database Rules.....	162

## C. Integration With ManageX

.....	164
Integration Possibilities.....	164
Installation.....	165
Service Desk Application Server.....	165
Demo Database.....	166
ManageX Application Server.....	166
ServiceDeskSamplePassThru.....	166
ManageX on a Different Server.....	167
Configuration.....	168
Configuring Service Desk.....	168
Configure the Ini File.....	168
Database Rules.....	168
ManageX Server Account.....	169
Configuring ManageX.....	170
Configure the Acceptance Filter.....	170
ManageX Lights Out Policy.....	172
The Event Queue.....	176

## D. Examples

Using an Example Configuration File.....	180
--	-----

Importing From an MS Excel Spreadsheet .....	183
Importing Data With Relations .....	193
Data Source .....	193
Preparing the Excel Sheet for Data Exchange .....	195
Defining the ODBC Link .....	196
Configuring the Extractor and the Import Mapping .....	199
Configuring the DSN Section .....	203
Configuring the SYSTEM Section .....	203
Configuring the CLASSES Section .....	203
Configuring the SOFTWARE Class .....	204
Import Mapping for the SOFTWARE Class .....	204
Configuring the ORG Class .....	207
Import Mapping for the ORG Class .....	207
Configuring the EMP Class .....	209
Import Mapping for the EMP Class .....	209
Configuring the PHONE Class .....	211
Import Mapping for the PHONE Class .....	212
Configuring the HARDWARE Class .....	214
Import Mapping for the HARDWARE Class .....	214
Configuring the HWUSERS Class .....	216
Import Mapping for the HWUSERS Class .....	216
Configuring the REL Class .....	218
Import Mapping for the REL Class .....	218
Configuring the WORKGROUP Class .....	221
Import Mapping for the WORKGROUP Class .....	221
Configuring the WORKGROUPEMP Class .....	223
Import Mapping for the WORKGROUPEMP Class .....	223
Importing the Data .....	225
Importing Multiple Telephone Numbers .....	226
Importing Data From an ASCII Text File .....	231

## Glossary

---

# Contents



---

# Figures

Figure 1-1. Overview of the Data Exchange Process . . . . .	28
Figure 2-1. Data Exchange Task Window . . . . .	51
Figure 2-2. Data Exchange dialog box - Exporting Data. . . . .	52
Figure 3-1. Import Mapping for PERSON Class . . . . .	65
Figure 3-2. Import Mapping for CI_ADMIN_RELATION Class . . . . .	66
Figure 3-3. Field Mapping for ADMIN_ID. . . . .	67
Figure 3-4. Import Mapping for CI_USER_RELATION_PARENT Class . . . . .	68
Figure 3-5. Import Mapping for CI_USER_RELATION_CHILD . . . . .	69
Figure 3-6. Field Mapping for Parent. . . . .	70
Figure 3-7. Import Mapping for CI_ID_NAME Class . . . . .	71
Figure 3-8. Import Mapping for SW_ID_NAME Class . . . . .	72
Figure 3-9. Import Mapping for SW_REL_RELATION . . . . .	73
Figure 3-10. Field Mapping for CL_ID. . . . .	74
Figure 3-11. Field Mapping for SW_ID . . . . .	74
Figure 3-12. Import Mapping dialog box for sd_event. . . . .	76
Figure 3-13. Import Mapping Task Window . . . . .	77
Figure 3-14. Import Mapping dialog box . . . . .	78
Figure 3-15. External Class dialog box . . . . .	80
Figure 3-16. Field Mapping dialog box. . . . .	80
Figure 3-17. Expanded Field Mapping dialog box . . . . .	82
Figure 4-1. Data Exchange Task dialog box . . . . .	87
Figure 4-2. Data Exchange dialog box - Importing Data. . . . .	88
Figure 4-3. Integration Account . . . . .	96

---

## Figures

Figure 5-1. Inbound Service Events . . . . .	100
Figure 6-1. Outbound Service Event Concept . . . . .	112
Figure 6-2. Select Database Item . . . . .	115
Figure 6-3. Set Database Condition . . . . .	116
Figure 6-4. Add Conditions . . . . .	117
Figure 6-5. Configure Conditions . . . . .	118
Figure 6-6. Add Actions . . . . .	119
Figure 6-7. Create a Command Exec Action. . . . .	120
Figure 6-8. Turn on the Rule. . . . .	122
Figure A-1. Modify an Event in NNM. . . . .	135
Figure A-2. Service Event Command Line in NNM. . . . .	136
Figure B-1. ITO Account dialog box . . . . .	151
Figure B-2. Add User dialog box . . . . .	156
Figure B-3. Configuration Item dialog box . . . . .	157
Figure C-1. ManageX Account dialog box. . . . .	170
Figure C-2. Set ManageX Acceptance Filters. . . . .	171
Figure C-3. ManageX ActiveX Script Action . . . . .	173
Figure D-1. Adjust the Excel Spreadsheet . . . . .	183
Figure D-2. ODBC Microsoft Excel Setup dialog box. . . . .	184
Figure D-3. New Import Mapping dialog box. . . . .	186
Figure D-4. New External Class dialog box . . . . .	187
Figure D-5. Mapping Attributes . . . . .	187
Figure D-6. Complete Import Mapping. . . . .	188

---

# Figures

Figure D-7. Export Data . . . . .	190
Figure D-8. Import Data . . . . .	192
Figure D-9. EMPLOYEE SOURCE . . . . .	194
Figure D-10. EMPLOYEE SOURCE - Continued . . . . .	194
Figure D-11. ORGANIZATION, SOFTWARE & WORKGROUP SOURCES . . . . .	195
Figure D-12. HARDWARE SOURCE . . . . .	195
Figure D-13. Excel File Name Definition . . . . .	196
Figure D-14. Create New ODBC Connection . . . . .	197
Figure D-15. New Data Source . . . . .	198
Figure D-16. Excel ODBC_DSN . . . . .	199
Figure D-17. Export Mapping in Data Exchange dialog box . . . . .	200
Figure D-18. Import Mapping - SOFTWARE . . . . .	206
Figure D-19. Mapping External Class - SOFTWARE . . . . .	207
Figure D-20. Import Mapping - ORG . . . . .	208
Figure D-21. Mapping External Class - ORG . . . . .	209
Figure D-22. Import Mapping - EMP . . . . .	210
Figure D-23. Mapping External Class - EMP . . . . .	211
Figure D-24. Import Mapping - PHONE . . . . .	213
Figure D-25. Mapping External Class - PHONE . . . . .	214
Figure D-26. Import Mapping - HARDWARE . . . . .	215
Figure D-27. Mapping External Class - HARDWARE . . . . .	216
Figure D-28. Import Mapping - HWUSERS . . . . .	217
Figure D-29. Mapping External Class - HWUSERS . . . . .	218

---

## Figures

Figure D-30. Import Mapping - REL. ....	220
Figure D-31. Mapping External Class - REL .....	221
Figure D-32. Import Mapping - WORKGROUP .....	222
Figure D-33. Mapping External Class - WORKGROUP .....	223
Figure D-34. Import Mapping - WORKGROUPEMP .....	224
Figure D-35. Mapping External Class - WORKGROUPEMP .....	225
Figure D-36. Import Mapping for CL_CONTACT .....	227
Figure D-37. Import Mapping CL_TEL_CONTACT1 .....	228
Figure D-38. Field Mapping - Person Field. ....	229
Figure D-39. Field Mapping - Number Field .....	229
Figure D-40. Field Mapping - Telephone Type .....	230
Figure D-41. ODBC Text Setup .....	232
Figure D-42. ODBC Text file - Import Mapping .....	234

Table 1. Revision History . . . . .	17
Table 3-1. Batch Loading . . . . .	60
Table 4-1. Complete Importing and Exporting Commands . . . . .	92
Table 4-2. Exporting Command Syntax Defined . . . . .	92
Table 4-3. Importing Command Syntax Defined . . . . .	93
Table 4-4. Importing and Exporting Command Syntax Defined . . . . .	94
Table 5-1. Service Event Switches . . . . .	101
Table 5-2. Service Event Configuration File . . . . .	103
Table 7-1. File Directory . . . . .	125
Table 7-2. Problem Solving . . . . .	126
Table 7-3. Tips . . . . .	128
Table A-1. Service Desk Server . . . . .	133
Table A-2. Sequential Attribute Variables . . . . .	138
Table A-3. Special Information Variables . . . . .	139
Table A-4. SNMP Specific Variables . . . . .	140
Table A-5. NNM UNIX Server . . . . .	142
Table B-1. Service Desk Server . . . . .	149
Table B-2. ITO server . . . . .	153
Table B-3. Service Desk Attributes . . . . .	159
Table C-1. Service Desk Server . . . . .	165
Table C-2. Service Desk Server . . . . .	166
Table C-3. ManageX Event String Substitutions . . . . .	174
Table C-4. ManageX Event COM Objects . . . . .	175

---

# Tables

---

## Preface

Service Desk is an integral part of the HP OpenView portfolio of products. The Service Desk application has a generic data exchange interface that facilitates open integration with third party applications for the purpose of sharing data. Data exchange is the process of exporting data from one application database and importing it into another, in a format that can be used by the application. Human resource data or inventory data can be exported from one application, reformatted and imported into Service Desk. Service events such as a single incident, employee record, or configuration item can also be inserted and updated, allowing you to tie all of your system management applications together.

The *HP OpenView Service Desk: Data Exchange Administrator's Guide* provides you with the ability to better understand the capabilities that exist for exporting data from external sources and importing data into Service Desk. This guide is intended for system administrators who will configure the data exchange settings, or for others involved with data exchange. It is assumed that users of this manual have an understanding of databases.

This guide contains both conceptual and detailed how-to information for configuring and using the data exchange features of Service Desk. A brief outline of the information in this guide is below:

- Chapter 1, “The Data Exchange Concept” on page 25, provides the reader with a conceptual overview of how the entire data exchange process works.
- Chapter 2, “Exporting Data” on page 33, explains how to configure an extractor, how to export data, and how to review exported XML files with the Viewer.
- Chapter 3, “The Import Mapping” on page 59, provides detailed descriptions of the import process with information on mapping classes and properties to items and attributes in Service Desk.
- Chapter 4, “Importing Data in Batches” on page 85, explains how to import batches of data from an external data source into Service Desk; for example, from a network management database into the Service Desk database.
- Chapter 5, “Importing Service Events” on page 99, provides information about the service event command line and explains how

to import service events.

- Chapter 6, “Outbound Service Events” on page 111, provides conceptual information on sending service events from Service Desk to an external application.
- Chapter 7, “Auditing and Troubleshooting” on page 123 provides some helpful information about the auditing and troubleshooting tools available with Data Exchange.
- Appendix A, “Integration With Network Node Manager” on page 131 contains useful information for importing service events from HP OpenView Network Node Manager into Service Desk.
- Appendix B, “Integration With ITO” on page 145 contains information useful for setting up a bi-directional service event integration with ITO.
- Appendix C, “Integration With ManageX” on page 163 contains information useful in setting up a bi-directional service event integration with HP OpenView ManageX.
- Appendix D, “Examples” on page 179 includes examples for modifying a configuration file to export data, importing data from a Microsoft® Excel spreadsheet, and from an ASCII text file.



## Revision History

When an edition of a manual is issued with a software release, it has been reviewed and tested and is therefore considered correct at the date of publication. However, errors in the software or documentation that were unknown at the time of release, or important new developments, may necessitate the release of a service pack that includes revised documentation. Revised documentation may also be published on the Internet, see the section We Welcome Your Comments! below, for the URL.

A revised edition will display change bars in the left-hand margin to indicate revised text. These change bars will only mark the text that has been edited or inserted since the previous edition or revised edition.

When a revised edition of this document is published, the latest revised edition nullifies all previous editions.

**Table 1**                      **Revision History**

<b>Edition and Revision Number</b>	<b>Issue Date</b>	<b>Product Release</b>
Second Edition	June 2000	Service Desk 3.0
Second Edition, 1st Revision	December 2000	Service Desk 3.0, Service Pack 3

## Related Publications

This section helps you find information that is related to the information in this guide. It gives an overview of the Service Desk documentation and lists other publications you may need to refer to when using this guide.

### The Service Desk Documentation

Service Desk provides a selection of books and online help to assist you in using Service Desk and improve your understanding of the underlying concepts. This section illustrates what information is available and where you can find it.

---

#### NOTE

This section lists the publications provided with Service Desk 3.0. Updates of publications and additional publications may be provided in later service packs. For an overview of the documentation provided in service packs, please refer to the readme file of the latest service pack. The service packs and the latest versions of publications are available on the Internet. See the section “We Welcome Your Comments!” in this preface for the URLs.

---

- The `Readme.htm` file on the Service Desk CD-ROM contains information that will help you get started with Service Desk. It also contains any last-minute information that became available after the other documentation went to manufacturing.
- The *HP OpenView Service Desk: Release Notes* give a description of the features that Service Desk provides. In addition, they give information that helps you:
  - compare the current software’s features with those available in previous versions of the software;
  - solve known problems.

The Release Notes are available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Release_Notes.pdf`.

- The *HP OpenView Service Desk: Supported Platforms List* contains information that helps you determine platform and software requirements and compatibility. It lists the combinations of platforms and software Service Desk 3.0 was tested on.

The Supported Platforms List is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Supported_Platforms_List.pdf`.

- The *HP OpenView Service Desk: Installation Guide* covers all aspects of installing Service Desk.

The Installation Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Installation_Guide.pdf`.

- The *HP OpenView Service Desk: Data Exchange Administrator's Guide* explains how you can use data from other applications in Service Desk. It explains the underlying concepts of the data exchange process and gives step-by-step instructions on exporting data from external applications and importing it into Service Desk. The data exchange process includes importing single service events and batches of data.

The Data Exchange Administrator's Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Data_Exchange.pdf`.

- The *HP OpenView Service Desk: API Programmer's Guide* contains information that will help you create customized integrations with Service Desk. This guide depicts the API structure, and explains some of the basic functions with examples for using the Application Programming Interface (API) provided with Service Desk. The API extends the HP OpenView Service Desk environment by providing independent programmatic access to data-centered functionality in the Service Desk application server environment.

The API Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `API_pg.pdf`.

- The *HP OpenView Service Desk: Data Dictionary* contains helpful information about the structure of the application.

The Data Dictionary is available as an HTML file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Data_Dictionary.htm`.

- The *HP OpenView VantagePoint Service Desk 3.0 Computer Based Training (CBT)* CD-ROM is intended to assist you in learning about the functionality of HP OpenView VantagePoint Service Desk 3.0 from both a user and a system administrator perspective. The CD-ROM contains demonstration videos and accompanying texts that

explain and show how to perform a wide variety of tasks within the application. The CBT also explains the basic concepts of the Service Desk application.

The CBT is shipped automatically with the regular Service Desk software on a separate CD-ROM.

- The online help is an extensive information system providing:
  - procedural information to help you perform tasks, whether you are a novice or an experienced user;
  - background and overview information to help you improve your understanding of the underlying concepts and structure of Service Desk;
  - information about error messages that may appear when working with Service Desk, together with information on solving these errors;
  - help on help to learn more about the online help.

The online help is automatically installed as part of the Service Desk application and can be invoked from within Service Desk. See the following section entitled “Using the Online Help” for more information.


### **Reading PDF Files**


You can view and print the PDF files with Adobe® Acrobat® Reader. This software is included on the HP OpenView Service Desk 3.0 CD-ROM. For installation instructions, see the `readme.htm` file on the CD-ROM.

The latest version of Adobe Acrobat Reader is also freely available from Adobe’s Internet site at <http://www.adobe.com>.

### **Using the Online Help**

You can invoke help from within Service Desk in the following ways:

- To get help for the window or dialog box you are working in, do one of the following:
  - Press **F1**.
  - Click the help toolbar button .
  - Choose **Help** from the **Help** menu.

— Click the help command button  in a dialog box.

- To search for help on a specific subject using the table of contents or the index of the help system: choose Help Contents & Index from the Help menu.



When you are in the help viewer, you can find help on how to use the help system itself by clicking the Help toolbar button:




Service Desk also provides *tooltips* and “What’s This?” help for screen items like buttons, boxes, and menus.

A *tooltip* is a short description of a screen item. To view a tooltip, rest the mouse pointer on the screen item. The tooltip will appear at the position of the mouse pointer.

“What’s This?” help is a brief explanation of how to use a screen item. “What’s This?” help generally gives more information than tooltips. To view “What’s This?” help:

1. First activate the “What’s This?” mouse pointer in one of the following ways:
  - Press **Shift+F1**.
  - Click the “What’s This?” toolbar button .
  - Choose What’s This? from the Help menu.
  - In dialog boxes, click the question mark button  in the title bar.

The mouse pointer changes to a “What’s This?” mouse pointer .

2. Then click the screen item for which you want information. The “What’s This?” help information appears in a pop-up window.

To close the pop-up window, click anywhere on the screen or press any key on your keyboard.

## Typographic Conventions

The table below illustrates the typographic conventions used in this guide.

Font	What the Font Represents	Example
<i>Italic</i>	References to book titles  Emphasized text	See also the <i>HP OpenView Service Desk: Installation Guide</i> .  <i>Do not delete</i> the System user.
<b>Bold</b>	First-time use of a term that is explained in the glossary	The <b>service call</b> is the basis for incident registration.
Courier	Menu names  Menu commands  Button names  File names  Computer-generated output, such as command lines and program listings	You can adjust the data view with the commands in the View menu.  Choose Save from the menu.  Click Add to open the Add Service Call dialog box.  To start the installation, double-click setup.htm.  If the system displays the text C:\>dir a: The device is not ready then check if the disk is placed in the disk drive.
<b>Courier bold</b>	User input: text that you must enter in a box or after a command line	If the service call must be solved within 30 minutes, enter <b>30</b> .
<i>Courier italic</i>	Replaceable text: text that you must replace by the text that is appropriate for your situation	Go to the folder X:\Setup, where X is your CD-ROM drive.
<b>Helvetica bold</b>	Keyboard keys  A plus sign (+) means you must press the first key ( <b>Ctrl</b> in the example), hold it, and then press the second key ( <b>F1</b> in the example).	Press <b>Ctrl+F1</b> .

## **We Welcome Your Comments!**

Your comments and suggestions help us understand your needs, and better meet them. We are interested in what you think of this manual and invite you to alert us to problems or suggest improvements. You can submit your comments through the Internet, using the HP OpenView Documentation Comments Web site at the following URL:

[http://ovweb.external.hp.com/lpe/comm\\_serv](http://ovweb.external.hp.com/lpe/comm_serv)

If you encounter *serious errors* that impair your ability to use the product, please contact the HP Response Center or your support representative.

The latest versions of OpenView product manuals, including Service Desk manuals, are available on the HP OpenView Manuals Web site at the following URL:

[http://ovweb.external.hp.com/lpe/doc\\_serv](http://ovweb.external.hp.com/lpe/doc_serv)

Software patches and documentation updates that occur after a product release, will be available on the HP OpenView Patches Web site at the following URL:

<http://ovweb.external.hp.com/cpe/patches>





---

# **1            The Data Exchange Concept**

Data Exchange is the process of exporting information from a data source, formatting it and then importing it into the Service Desk application. A number of example configurable extractors are provided, that can be used to gather data from multiple data sources. The extractor

exports a data exchange file in extensible markup language meeting the common information model, (CIM-XML) making it easy to import. Mapping the import settings in Service Desk ensures the data is imported correctly.

The extractor will work with any third-party database with ODBC, extending the capability of Service Desk even further.

The following chapters provide more detailed information about the extractor and the data exchange process.

---

**NOTE**

For additional information on installing components necessary for data exchange, see the *HP OpenView Service Desk: Installation Guide*.

---

## The Big Picture

The command to exchange data is sent from Service Desk. Data is exported from the external application's database by an extractor configured for the application. The key steps involved in the process are:

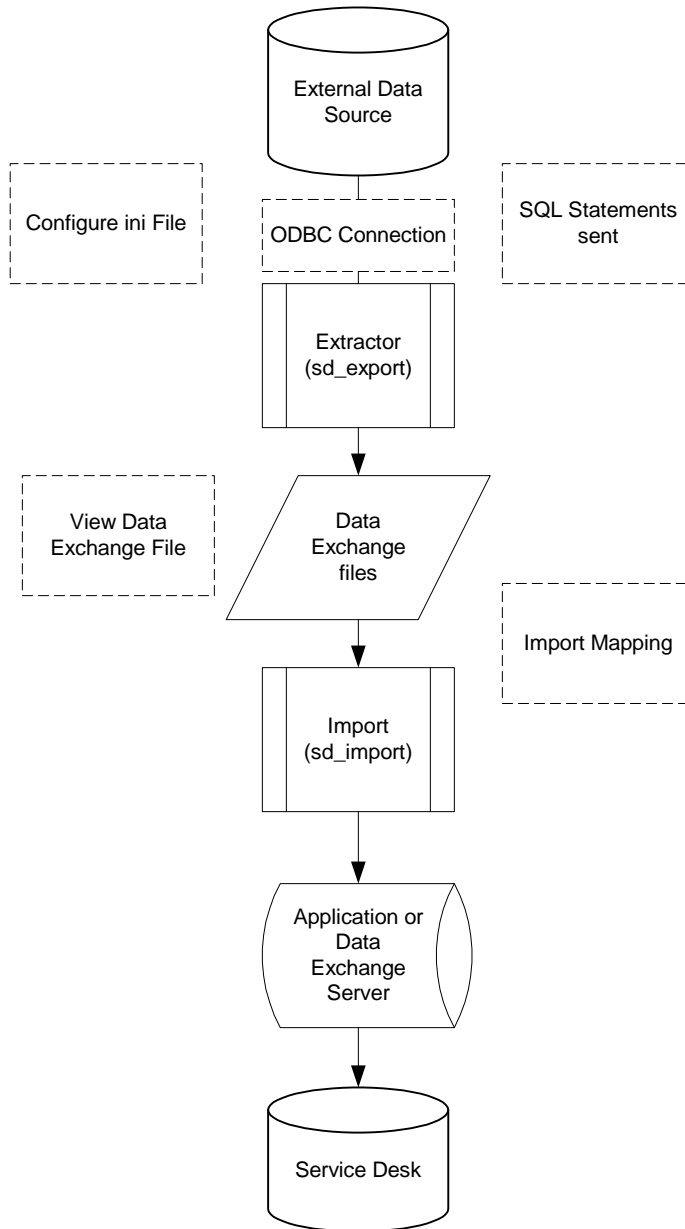
- Step 1.** Create or modify the configuration file using one of the examples provided.
- Step 2.** Establish an ODBC link to the external application's database.
- Step 3.** Export the data by running the extractor from the Service Desk application server or a dedicated data exchange server.

Examine the exported XML file with the viewer, if needed configure the extractor and export the data again

- Step 4.** Configure the data import settings by mapping the external application's classes, properties and values to Service Desk items, attributes, and values,
- Step 5.** Import the data from the XML file into Service Desk. Batch importing is done from the Data Exchange dialog box, see Chapter 4, "Importing Data in Batches," on page 85. The data exchange process for service events is initiated from a command line in the external application, see Chapter 5, "Importing Service Events," on page 99.

Once the data exchange process is configured for an external application, you can save the settings as a task and use the task to export and import data again. It is also possible to use a scheduling program, for more information see "Scheduling a Data Exchange Task" on page 91

**Figure 1-1 Overview of the Data Exchange Process**



---

**NOTE**

The data exchange process only works if Service Desk runs in three-tier mode. The data exchange process can only be run from an application server or a data exchange server running on a Windows NT® platform. Configuration, XML, and log files can only be accessed from the application server or the data exchange server.

---

## Overview of the Export Process

Most companies use a number of applications to monitor their networks, store human resource data, and inventory information. This information can be imported into Service Desk using Data Exchange. To export data from the external application's database, a special program called an extractor is needed. The extractor has two major tasks. First, to export the information that needs to be imported into Service Desk. Second, to configure the exported information into a format that can be imported properly by Service Desk.

System administrators can create new configuration files by copying one of the examples provided and modifying it. The configuration file makes it possible to establish what data to gather from the external application's database. Changing the configurable file changes the export mapping, giving system administrators the ability to vary the data that is exported into the data exchange files.

When the extractor receives the Export data from a storage device command, it connects to the database of the external application via the ODBC driver and reads only the data it is configured to read.

The following types of information can be defined with the configuration file:

- Database information
- System information
- Element information

The exported data is formatted into an XML file, referred to as a data exchange file, that can be viewed prior to importing into Service Desk. The data exchange file is then imported into the Service Desk database so that the data can be displayed in the Service Desk application.

For more detailed information See "Exporting Data" on page 33.

## **Configuring the Import Mapping**

It is important to configure the import mapping so that the data imported from another application is interpreted correctly. This process is completed through the use of import mapping dialog boxes in the Service Desk application. To complete this task, it is necessary to know what information is in the external application database and where and how the data must be displayed in Service Desk. Classes will need to be mapped to a template which contains default values. Templates exist for every item available in Service Desk. Attributes, relations, and attribute values will also need to be mapped if you intend to import them into Service Desk.

For more detailed information, See “The Import Mapping” on page 59.

## Importing the Data

Once the import mapping is configured and the configurable extractor is configured to export data from your particular external application, it is possible to start importing data into Service Desk. There are two options available for the purpose of importing data; importing batches of data with `sd_import.exe` or importing service events with `sd_event.exe`.

The batch import process makes it possible to import a number of different items. The data is organized into a preload file; data that cannot be mapped will be filtered out. The filtered data in the preload file is then imported.

The option to import a service event is recommended for use when there is only one class or configuration item to be imported, for example if you only want to import a service call, or an incident. A command is entered, in the external application, to send data to the Service Desk application server where it is imported. While the data exchange process can only be run on a Windows NT server, the command to import a service event can come from any machine that uses hypertext transfer protocol (HTTP) for Web servers, to include UNIX® machines.

The import mapping defined in Service Desk is used by both the batch and service event importing options. The import mapping used for a service event is selected from the command line, while the import mapping for batch imports is selected in the data exchange dialog box in the Service Desk application. Templates exist for both types of import mapping.

All Data Exchange processes take place through the application server; if you will be importing a large number of items it will be necessary to use a dedicated data exchange server.

---

### CAUTION

Old objects and relations are not removed in this version of Service Desk. You must manually locate and delete old objects and relations to remove them.

---

For more detailed information See “Importing Data in Batches” on page 85.



---

## **2** **Exporting Data**

To import data into Service Desk it first needs to be exported from the external data source. A number of example extractors are provided with Service Desk that can be configured for your data source.

The extractor's purpose is to take data from where it is stored by one application and put it in a format that can be imported by another application.

The export mapping is set in the configuration file so that it will export only specific data. Any number of classes and properties can be exported. The information is exported into XML formatted files that are compliant with the common information model, also known as CIM-XML. This format is used because it can be read and imported by many different database applications without needing to change the extractor. A DTD file, located in the same folder as your XML file, is used to validate the file, check the tag hierarchy and tag spelling. The file will also be checked for misspelled class and property names via the import mapping, and whether relations reference objects in the exported XML file or not. A text file and a log file are also created by the extractor.

The entire data export process includes approximately 4 steps, which will be explained in the following sections. They are:

- Step 1.** Decide what to export.
- Step 2.** Configure the extractor, using one of the example configuration files provided.
- Step 3.** Define an ODBC link to the external source database.
- Step 4.** Run the extractor to export CIM-XML files.
- Step 5.** View the XML files.

## Deciding What to Export

Before configuring the extractor and setting the export mapping to export data, you need to know what information you want to manage or view in Service Desk. Many organizations use Service Desk in conjunction with other management applications and want to avoid adding inventory or human resource information manually into Service Desk. The following list is a guide to help you determine what to configure the extractor to export:

1. Define the information you want to manage in Service Desk.
2. Evaluate the Service Desk database to see how information is stored. This should result in a list of table names, column names and attributes.
3. Select the items you want to export from the external data source.
4. Evaluate the external application database: Can it provide the information you selected or will you need to use an additional management application? You should develop a list of class names, attributes, key values and any relations. You will need to note how the data is organized in the database. The following section may be helpful in this task.

---

### TIP

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

---

## Checking the Database Structure

One of the most important steps in creating or modifying a configurable extractor file is knowing the table structure. Some applications do not have an object browse tool and it can be difficult to view how information in the external application database is structured. It is crucial that table names and columns are accurate when configuring the extractor. For Oracle® database users it is possible to use SQL\*Plus® for this purpose. You may also use the technical documentation provided with the external application. Another possible method is to use Microsoft Access as explained below to browse through the data and make simple queries:

Exporting Data  
**Deciding What to Export**

1. **Start MS Access.**
2. **Create New, Blank Database.**
3. **Enter a Name for testing purposes and click Create.**
4. **From the File menu click Get External Data, then click Link Tables.**
5. **Select File of Type ODBC databases.**
6. **Click the Machine Data Source tab.**
7. **Select the ODBC driver.**
8. **Select the tables you want to add and click OK.**
9. **Click Open to view the contents, or Design to view the columns of a table. Use Query building to browse data and links.**

## Configuring the Extractor

The extractor comes with a number of example extractors. The examples contain files in Windows® initialization format that can be configured. Windows initialization files are used to define and transmit commands. System administrators can use the configuration file to specify precisely what data they want to export and how and where it needs to be exported from the database. The examples provided contain default information and can be used as is or modified. The parameters set in the configuration file are directions sent to the external application database through SQL statements. The SQL statements are executed and the information is exported in the format defined.

When configuring the import settings, keep in mind:

- The class name between [ ] brackets in the configuration file is the class you need to map in the import mapping.
- ATT and PARENT\_RELATION\_NAME determine the attributes that have to be mapped when configuring the import mapping.

## Modifying the Configuration File

A number of configuration files in Windows ini format are provided to make it easier for system administrators to make adjustments concerning the data to be exported. Example configuration files are provided for: Microsoft Systems Management Server , HP OpenView's Network Node Manager , and Desktop Administrator. To view the files look in the `data_exchange\config` folder of the Service Desk application or open them from the Data Exchange dialog box from within the Administrator Console. A portion of a configuration file follows with the keywords defined in the following section:

### Example 2-1

#### Configuration File

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

## Exporting Data

### Configuring the Extractor

```
[ SYSTEM ]
LOG=TRUE
XML=TRUE
TXT=TRUE
DUMP=TRUE
OUTPUT_FILE=dta5.txt
LOG_FILE=C:\Temp\DTA5.log
XML_OUTPUT_FILE=C:\Temp\DTA5.xml
APPLICATION_NAME=DTA5

ENCODING=ISO-8859-1

[ CLASSES ]
NAME=ADMIN,CPU

[ ADMIN ]
SOURCE=ADMINISTRATIVE
ID=[MACHID]
ATT=[MACHINENAME],[IPADDRESS]
CONDITION=[OSTYPE]='Windows NT' OR [OSTYPE]='Windows 95'
COLUMNS=[ADMINISTRATIVE].[MACHINEID] as
[MACHID],[ADMINISTRATIVE].[MACHINENAME] as
[MACHINENAME],[ADMINISTRATIVE].[IPADDRESS] as [IPADDRESS]

[ 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]

## Configuration File Keywords

Definitions of the syntax used in the configuration file follow:

<b>DSN</b>	<b>Define the data source to be used</b>
NAME	Name of the ODBC data source.
USR	The database user who owns the data source tables and views
PWD	The database user's password
<b>SYSTEM</b>	<b>Define the system and and data file settings</b>
LOG	Enter TRUE to create a log file created.
TXT	Enter TRUE to create a text file.
XML	Enter TRUE to create an XML file.
DUMP	Enter TRUE to have the output displayed on your screen.
LOG_FILE	Enter the location where you want the log file placed. This field is only used when you run Data Exchange from a command line.
XML_OUTPUT_FILE	Enter where you want the extracted XML file placed. This field is only used when you run Data Exchange from a command line.
APPLICATION_NAME	Enter the name of the external application you are extracting data from. For example, DTA.
ENCODING	Enter the name of the language code you want to use when viewing the XML file. For example, the default ISO-8859-1 is used for European languages while UTF-8 is used for Kanji and some other character based languages.

<b>CLASSES</b>	<b>Define what entities will be exported</b>
NAME	Name of the item you will be importing. You can choose the name yourself. This is the class name that is mapped from the XML file to a Service Desk item when you do the import mapping.
SOURCE	Specify the tables or views to select data from.
ID	Only used in text output file, for backwards compatibility with ITSM.
ATT	Defines the contents of the classes. Alias names of attributes you want to export to XML file. Not all columns specified in COLUMNS have to appear in the XML file.
COLUMNS	Specifies all columns to be selected with an alias [columnname1] AS [alias 1]. Columns that appear in the PARENT_RELATION have to be put into the column list when tables are cached [LOADTABLE=TRUE], because the extractor uses this information to find all children for a particular parent in memory. If an alias is not specified the column name will be used as the alias name.
MAXRECORDS	Use to specify the maximum number of records to be exported.
LOADTABLE	TRUE or FALSE. Specifies whether the records are cached in memory to process parent-child relations faster, or queries are run for each parent to find its children. Use TRUE if you have plenty of memory and want quicker performance. Use FALSE if you have little memory available, it will be slower.
PARENT	Specifies the parent class in the child section.
PARENT_RELATION	Specifies the relation between the child and its parent. Left of the equal sign is the column of the foreign key of the child. Right of the equal sign is the alias of the primary key of the parent.



<b>CLASSES</b>	<b>Define what entities will be exported</b>
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.

---

**NOTE**

Do not use ID as one of your field names in the configuration file, it will cause an error when exporting. When `sd_export` extracts data and puts it into XML files a field called ID is added by the program to give each record a unique ID in the XML file. As an alternative you can use field names such as `NNM_ID`, or `Event_ID`, for example.

---

## Defining Relations in the Configuration File

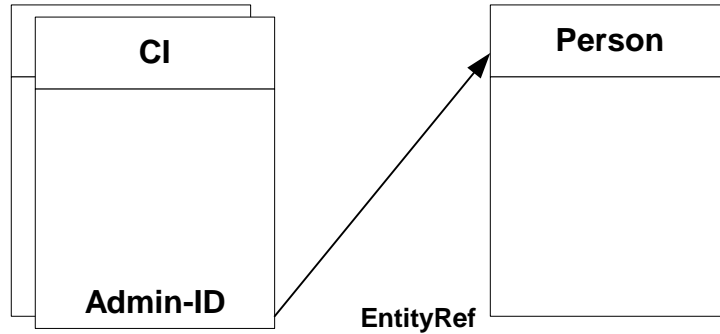
You can specify how relations should be imported in the export configuration ini file. The ini file is also where you specify how data is joined and sorted, not during the import mapping phase. There are three primary types of relations that can be imported; N:1 relations based on a search key, N:N or Parent-Child relations, and 1:Rel:1 that involve multiple classes. The following sections will explain the relation types in more detail and include information on how to configure the configurable ini file to export them.

The Data Dictionary provided with Service Desk provides a description for all items and tables in Service Desk to include the relations between those items. For example, if the attribute for an item has a 1:1 relation, e.g. Service call caller to the Person item, the table where the other item is stored will be listed. If the attribute has a 1:N relation, e.g. Service call History lines to History line items, the table where the items are stored will be shown along with the table column that contains the foreign key.

### Search Key; N:1 Relations

In the N:1 relation the N refers to one or more items in Service Desk related to one (1) other item in Service Desk. This type of relation uses a search code as an entity reference to determine the relationship between

the items. When modifying the configuration file you need to make sure that the item 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 will result when importing. The following diagram shows this relation with the Admin CI referencing the Person item:



Following is a portion of a configuration file demonstrating how to export this type of relation, note that PERSON is configured to be exported before CI\_ADMIN\_RELATION:

```
[PERSON]
SOURCE=[EMPLOYEE]
ATT=[PERSON_ID], [PERSON_NAME]
COLUMNS=[PERSON_ID] , [PERSON_NAME]
LOADTABLE= TRUE

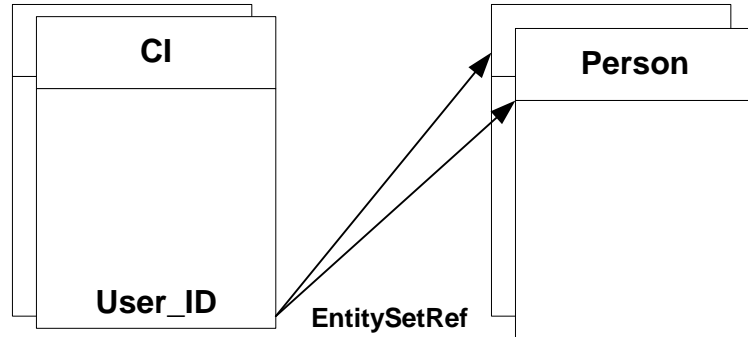
[CI_ADMIN_RELATION]
SOURCE=[EQUIPMENT]
ATT=[ADMIN_ID], [CI_ID]
COLUMNS=[ADMIN_ID],[EQUIPMENT_ID] AS [CI_ID]
LOADTABLE=TRUE
```

To complete the process for importing this relation you will need to create the import mapping to use a search code as a reference to the item. A complete import mapping for this example is available at; “Search Key; N:1 Relations” on page 64.

For another importing example see “Importing Data With Relations” on page 193.

## Parent-Child; N:N Relations

In this type of relation the N refers to one or more items in Service Desk related to N, one or more other items in Service Desk. This type of relation uses multiple item search codes or an entity set reference to make the relationship. The configuration file is modified differently for a N:N relation than a 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. The following diagram demonstrates this relation with multiple configuration items. Users are related to Person items using the EntitySetRef. The relationship can be made in either direction:



Following is a portion of an ini file demonstrating how to export this type of relation:

```
[CI_USER_RELATION_PARENT]
SOURCE=[EQUIPEMENT]
ATT=[USER_ID]
COLUMNS=[USER_ID],[EQUIPMENT_ID]AS[CI_ID]
LOADTABLE=TRUE

[CI_USER_RELATION_CHILD]
SOURCE=[EQUIPEMENT]
ATT=[CI_ID]
COLUMNS=[EQUIPMENT_ID]AS[CI_ID]
LOADTABLE=TRUE
PARENT=CI_USER_RELATION_PARENT
PARENT_RELATION=[EQUIPMENT_ID]=[CI_ID]
PARENT_RELATION_NAME=PARENT
```

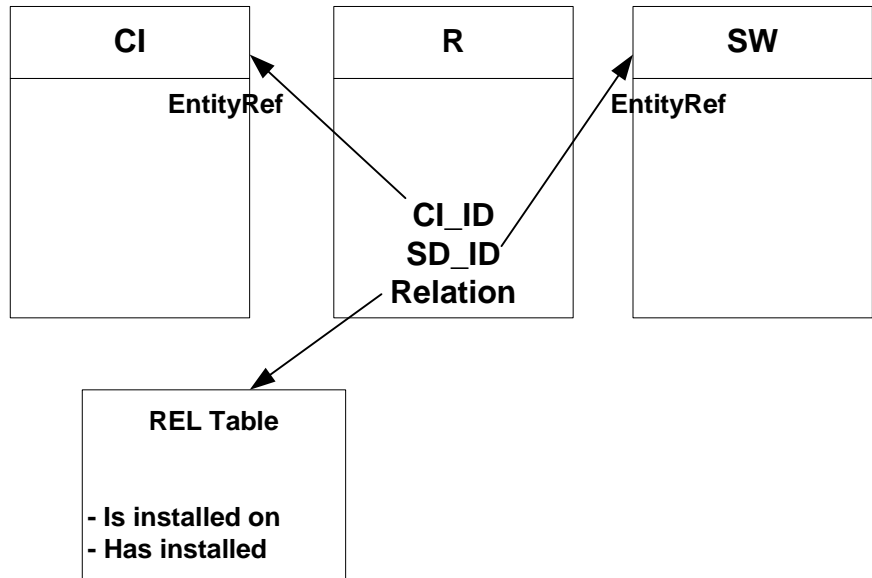
To finish the process and import this relation you will need to create an entity reference in the import mapping.

To complete the process for importing this relation you will need to create the import mapping to use a search code as a reference to the item. The complete import mapping for this example can be found at, "Parent-Child; N:N Relations" on page 67.

For another importing example see "Importing Data With Relations" on page 193.

### Relations with Relation Type; 1:Rel:1 Relations

The 1:Rel:1 type relation is a relation that is specified by creating another 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 Service Desk 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 items to the values in the relation table to define the type of relation. Another option is to enter a default value or create value mapping for the relations.



Following is a portion of an ini file demonstrating how to export this type

of relation:

```
[ CI_ID_NAME ]  
SOURCE=[ EQUIPMENT ]  
ATT=[ CI_ID ]  
COLUMNS=[ EQUIPMENT_ID ]AS[ CI_ID ]  
LOADTABLE=TRUE
```

```
[ SW_ID_NAME ]  
SOURCE=[ EQUIPMENT ]  
ATT=[ SW_ID ]  
COLUMNS=[ SW_ID ]  
LOADTABLE=TRUE
```

```
[ SW_REL_RELATION ]  
SOURCE=[ EQUIPMENT ]  
ATT=[ CI_ID ], [ SW_ID ]  
COLUMNS=[ EQUIPMENT_ID ]AS[ CI_ID ], [ SOFTWARE ]. [ SW_ID ],  
LOADTABLE=TRUE
```

To complete the process for importing this relation you will need to create the import mapping to use a search code as a reference to the item. Refer to the following section for the complete import mapping, “Relations with Relation Type; 1:Rel:1 Relations” on page 70.

For another importing example see “Importing Data With Relations” on page 193.

## Defining the ODBC link

An open database connectivity driver (ODBC) allows applications to communicate with a number of other ODBC compliant databases. To establish an ODBC link between the extractor and the external application database follow these general steps:

- Step 1.** Click **Start** on your desktop, then **Settings**, then click **Control Panel** and then the **ODBC** icon.
- Step 2.** Click the **System DSN** tab. **Service Desk** uses the **System DSN for Data Exchange**, and all examples are provided with that type of DSN in mind. **System DSN** is used when the data source is local to a computer, rather than dedicated to a user. The system, or any user with access privileges, can use a data source set up with a system DSN.
- Step 3.** Choose the appropriate driver for your application, for example:
  - DTA 5.0: Solid ODBCdriver 3.0
  - DTA 4.0: DTA 4.0 ODBC Driver (32-bit) or DTA 5.0 ODBC Driver.
  - Microsoft Systems Management Server: SQL Server
  - NNM: Oracle 8, or Oracle 7.3 Version 2.5, or SQL server
- Step 4.** Select **Finish**.
- Step 5.** Fill in the following Fields, dependent upon the application:

*DTA 5:*

Data Source Name:HPDTADB\_DTA

Description: DSN for HP DTA SOLID database

Network Name: TCPIP <server name> 1313

If your DTA 5 server is different from the Service Desk server you still need to have the SOLID ODBC driver installed on the Service Desk application server.

*DTA 4:*

Data Source Name, example: my\_dta\_source

Select Data File, example: F:\HPDTA\DATA\NADMIN.DBD

*Microsoft Systems Management Server:*

Data Source Name, example: my\_sms\_source.

Description, example: Microsoft Systems Management Server

Server, example: Microsoft Systems Management Server\_server

Press Next the select SQL server authentication (see your Microsoft Systems Management Server system administrator if you don't know).

Login ID:

Password:

Click Next,

Change the default database to the database you want to connect to,

Click Next,

Click Finish.

Network Node Manager:

Data Source Name, example: NNM6

Description, example: Network Node Manager

SQL\*Net connect string, example: nnm\_server

User ID: ovdb

## Special Instructions for Network Node Manager

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

Follow one of the following procedures to export NNM version 6 topology data to an Oracle or SQL Server database:

For NNM on Windows NT:

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

USAGE:

```
ovdwconfig.ovpl [-rdb ODBC datasource] [-u user]
[-password password] [-type embedded|msSqlSrvr/oracle] [-load]
[-reload] [-env envVar=value]
```

For example, for an Oracle database:

USAGE:

```
ovdwconfig.ovpl -rdb NNM6 -u ovdb -password ovdb -type oracle -load
```

For example, for a SQL database:

## Exporting Data

### Defining the ODBC link

```
ovdwconfig.ovp-1 -rdb nnm6sql -u ovdb -password ovdb -type  
msSqlSrvr -load
```

4. Run `ovdwtopo -export -rdb nnm`. This will put the topology data in the Oracle account via the ODBC connection.

For NNM on HP-UX for Oracle:

1. Create an Oracle account called `ovdb` with the password `ovdb`.
2. Shut down Oracle or close all Oracle client processes:

```
$cd ORACLE_HOME/rdbms/lib  
$ /usr/bin/make -f  
/etc/opt/OV/share/conf/analysis/clntsh.mk
```
3. Create tables for the NNM data with:

```
#ovdwconfig.ovpl -rdb OOracle -u ovdb -password ovdb  
-load -type oracle
```
4. Load the topology into the Oracle `ovdb` account, with:

```
# ovdwtopo -export -rdb OOracle
```



## The Extraction Process

All data exchange processes must be run from either the application server or a dedicated data exchange server. The option exists to export data in one step and then import it at a later time or export the data and import it immediately afterwards.

SQL statements are generated by class definitions established in the configuration file and are passed to the ODBC driver. Tables, columns, and classes are selected, and relationships made according to the definitions in the configuration file. The extractor processes the SQL statements in the following manner:

**Step 1.** Starts with the first class without parents.

**Step 2.** Creates an SQL statement.

**Step 3.** For each record:

- a. Extracts the first record
- b. Puts it in the XML file.

**Step 4.** For all child classes:

- a. If `LOADTABLE=TRUE` or if no records are loaded, creates an SQL statement to load all records in memory and uses the `PARENT_RELATION` to scan through the records in memory to find the children.

else

- b. Creates an SQL statement to get the records belonging to the parent using the `PARENT_RELATION`.

**Step 5.** Processes all child records recursively.

**Step 6.** The following records are created from the class section:

```
table loaded LOADTABLE=TRUE or no child class:  
SELECT <COLUMNS>  
FROM <SOURCE>  
WHERE <CONDITON>  
ORDER BY <ORDERBY>
```

Exporting Data  
The Extraction Process

Table not loaded *LOADTABLE=FALSE* and it is a child class *PARENT=*:  
SELECT <COLUMNS>  
FROM <SOURCE>  
WHERE <CONDITON>AND<PARENT\_RELATION(with the right side  
filled in.)  
ORDER BY <ORDERBY>;

joins (LEFT JOIN, etc.)  
SELECT <COLUMNS>  
FROM <SOURCE>ON<CONDITION>  
ORDER BY<ORDERBY>

## Exporting Data

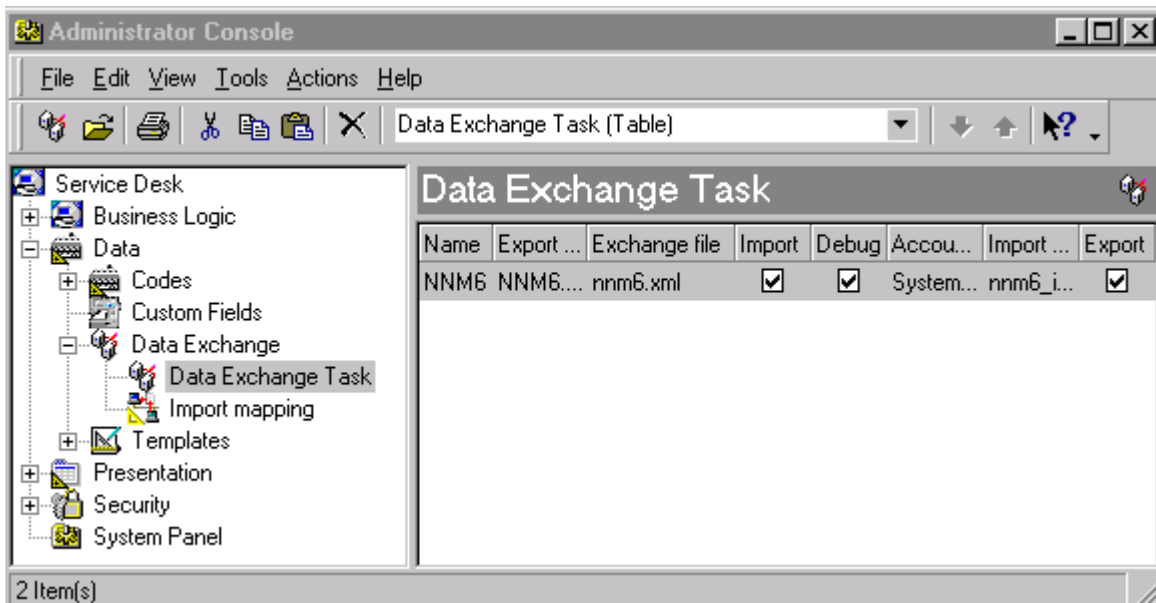
To export data follow the steps in the following procedure.

### Using a Task to Export Data From a Storage Device

**Step 1.** Open a data exchange task.

1. From the **Tools** menu click **System**, open the **Data** folder in the Administrator Console and double-click the **Data Exchange** icon to open it.
2. Double-click the **Data Exchange Tasks** icon. Existing data exchange tasks will be visible in the window. To export the task needs to include the correct: **Export** mapping, **Exchange** file name, and the **Export** check box must be selected:

**Figure 2-1 Data Exchange Task Window**



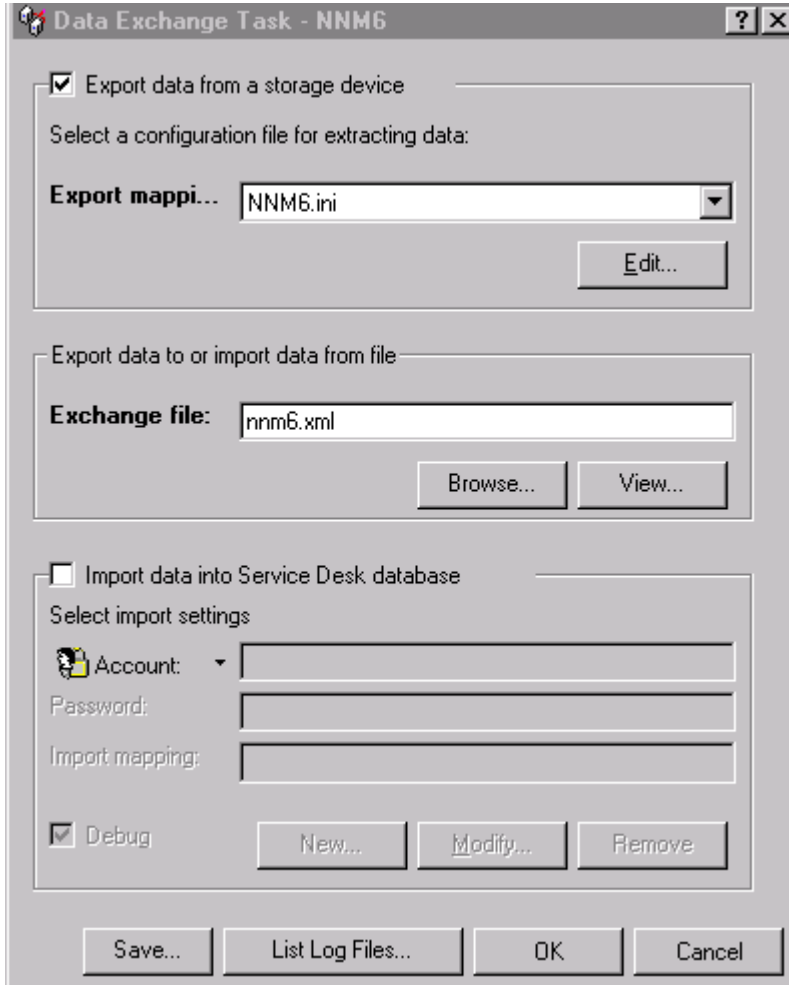
3. Double-click the task you want to use. The task will be opened in the Data Exchange dialog box.

Exporting Data  
Exporting Data

**Step 2.** Configure the task for exporting.

1. Select the `Export data from a storage device` check box:

**Figure 2-2** Data Exchange dialog box - Exporting Data



2. In the `Export mapping` field enter the configuration file you want to use with the extractor to export the data. You can choose one from the list box. Click `Edit` if you want to modify the configuration file selected. The file will be opened in a text editor.

3. In the `Exchange file` field, enter the name of the exported XML file you want to import after the export process has taken place. The XML file name is set in the extractor's configuration file. Click `Browse` to search for a file. Click the `View` button to open the exported XML file in an object-tree format to verify that it is accurate.

**Step 3.** Run the task.

1. Clear the `Import data into Service Desk database` check box and the `Account` field if you do not want to import data at this time. For information on importing data, see Chapter 4, "Importing Data in Batches," on page 85.
2. Click `Save` to save the modified task , and `OK` to export data.
3. To view log files of the export process click `List log files` at any time during or after the data exchange. Select a log file from the list, right-click and then click `Open`.

## Creating a Task to Export Data

To create a new task for exporting data:

1. If a task does not yet exist you can modify an existing task or you can create a new task by right-clicking with the mouse anywhere in the `Data Exchange Task` window and selecting `New Task` from the menu that will appear. Another option is to select the task button on the toolbar. It is located directly below the `File` button:
2. An empty data exchange dialog box will appear. Enter the information you want to use for this task and click `Save` at the bottom of the dialog box. The task will appear in the `Data Exchange Task` window.

## Exporting From the Command Line

To export data using a command line instead of the `Data Exchange` dialog see "Command Line Parameters" on page 92.

## The Data Exchange Files

The extractor creates a CIM-XML formatted file, a text formatted file, and a log file. Any XML file can be imported into `Service Desk` as long as it complies with the CIM-DTD. Currently CIM-DTD version 2.0 is

supported.

Each CIM-XML file contains a header with information in it such as the date and the application the data was exported from. Start and end tags are used to specify elements and properties within the file. Element tags are labeled INSTANCE CLASSNAME with the name for the item after it. The tag labeled PROPERTY denotes attributes of the INSTANCE. Each INSTANCE must have a PROPERTY named ID. The PROPERTY called ID must have a unique VALUE that is used to identify that specific item, such as a serial number. INSTANCES that are children will contain the tag PROPERTY.REFERENCE NAME =PARENT followed by values expressing the unique ID of the parent.

---

**NOTE**

The CIM\_DTD\_V20.dtd file must be located in the same folder as the generated XML file.

---

### The XML Format

Extensible markup language (XML) is similar to the HTML used in many Web pages. It provides a flexible way of creating common information formats making it a simple way to share the format and data across to other applications. Both XML and HTML contain markup symbols to describe the contents of a page or file. HTML, however, describes the content (mainly text and graphic images) only in terms of how it is to be displayed and interacted with. For example, a <P> starts a new paragraph. XML describes the content in terms of what data is being described. For example, a <PHONENUM> could indicate that the data that followed it was a phone number. This makes it possible for an XML file to be processed purely as data by an application. For example, depending on how the program in the receiving computer wanted to handle the phone number, it could be stored, displayed, or dialed. XML is "extensible" because, unlike HTML, the markup symbols are unlimited and self-defining.

### Example 2-2

#### Example XML File

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<!DOCTYPE CIM SYSTEM
"file:CIM_DTD_V20.dtd"

[<!ENTITY lt      "&#38;#60;">
```

```

<!ENTITY gt      "&#62;">
<!ENTITY amp     "&#38;#38;">
<!ENTITY apos    "&#39;">
<!ENTITY quot    "&#34;"> ]>
<CIM CIMVERSION="2.0" DTDVERSION="2.2">
<DECLARATION>
<DECLGROUP>

<DECLARATION>
<DECLGROUP>
  <VALUE.OBJECT>
    <INSTANCE CLASSNAME="Header">
      <PROPERTY NAME="Date" TYPE="string">
        <VALUE>12/06/1999</VALUE>
      </PROPERTY>
      <PROPERTY NAME="Application" TYPE="string">
        <VALUE>NNM</VALUE>
      </PROPERTY>
    </INSTANCE>
  </VALUE.OBJECT>
  <VALUE.OBJECT>
    <INSTANCE CLASSNAME="PC">
      <PROPERTY NAME="ID" TYPE="string">
        <VALUE>1</VALUE>
      </PROPERTY>
      <PROPERTY NAME="Name" TYPE="string">
        <VALUE>Vectra</VALUE>
      </PROPERTY>
      <PROPERTY NAME="Price" TYPE="real64">
        <VALUE>3200.00</VALUE>
      </PROPERTY>
      <PROPERTY NAME="Brand" TYPE="string">
        <VALUE>Hewlett-Packard</VALUE>
      </PROPERTY>
      <PROPERTY.REFERENCE NAME="User">
        <VALUE.REFERENCE>
          <INSTANCENAME CLASSNAME="Employee">
            <KEYBINDING NAME="ID">
              <KEYVALUE>4</KEYVALUE>

```

## Exporting Data

### Exporting Data

```
</KEYBINDING>
</INSTANCENAME>
</VALUE.REFERENCE>
</PROPERTY.REFERENCE>
</INSTANCE>
</VALUE.OBJECT>
<VALUE.OBJECT>
  <INSTANCE CLASSNAME="Mouse">
    <PROPERTY NAME="Name" TYPE="string">
      <VALUE>Cordless Pro V</VALUE>
    </PROPERTY>
    <PROPERTY NAME="Brand" TYPE="string">
      <VALUE>Hewlett-Packard</VALUE>
    </PROPERTY>
    <PROPERTY.REFERENCE NAME="Parent">
      <VALUE.REFERENCE>
        <INSTANCENAME CLASSNAME="PC">
          <KEYBINDING NAME="ID">
            <KEYVALUE>1</KEYVALUE>
          </KEYBINDING>
        </INSTANCENAME>
      </VALUE.REFERENCE>
    </PROPERTY.REFERENCE>
  </INSTANCE>
</VALUE.OBJECT>
```



## Viewing Data Exchange Files

Data exchange files can be viewed in a browser such as Internet Explorer. When the `View` button is selected the XML file is converted to HTML and structured into an object-tree format, making it easy to view. The viewer provides system administrators with a great tool for ensuring the extractor is configured correctly, and that the data exported is sufficient, prior to loading it into the Service Desk application database.

To view data exchange files prior to importing:

1. From the `Tools` menu, click `System`, then expand the `Data` folder in the `Administrator Console`. Double-click on the `Data Exchange` icon and then double-click the `Data Exchange Tasks` icon to open it.
2. Double-click on the task that corresponds with the exchange file you want to view. The task will be opened in the `Data Exchange` dialog box.
3. Enter the XML file you want to view in the `Exchange file` field, then click `View`.
4. When you finish viewing the file you can import it or run the export process again before importing.

---

### NOTE

Viewing large XML files with the Data Exchange Viewer is not recommended. A normal text editor is recommended when viewing larger XML files.

---

Exporting Data  
**Viewing Data Exchange Files**

---

## **3 The Import Mapping**

Import mapping is done via a series of mapping screens in the Service Desk application. Classes from the external application must be mapped to Service Desk items and provided with an appropriate template in Service Desk. The template selected comes with a number of default

attributes and values. Every item in Service Desk has a template which can be used for import mapping. Service Desk can be customized, making it possible to create new fields in Service Desk to support additional value mapping.

It can be helpful to use the viewer provided to look at the exported XML file when you are conducting the mapping process. This will give you an overview of all of the items you need to map. It can also help prevent minor deviations in terminology and spelling, that could lead to data not being imported.

The *HP OpenView Service Desk:Data Dictionary* contains helpful information about the structure of the Service Desk application. It is available as an HTML file on the Service Desk 3.0 CD-ROM with the file name `Data_Dictionary.htm`.

---

**NOTE**

External classes and attributes entered when configuring the import mapping must match exactly with those present in the XML file. If the item is in uppercase instead of lowercase letters, extra spaces are present, or the spelling differs a warning will be logged in the import log file during the import process.

---

The demo database contains predefined import mappings that can be changed to fit your organizational needs. For information about installing the demo database, see the *HP OpenView Service Desk: Installation Guide*. The following files are provided:

**Table 3-1 Batch Loading**

<b>Supported Integrations</b>	<b>Configurable Extractors</b>	<b>Import Mapping Templates</b>
Microsoft System Management Server Software	<code>sms2.ini</code>	<code>sms2_import</code>
HP OpenView Desktop Administrator	<code>dta5.ini</code>	<code>dta5_import</code>
HP OpenView Network Node Manager	<code>nnm6.ini</code>	<code>nnm6_import</code>

**Table 3-1**      **Batch Loading**

<b>Supported Integrations</b>	<b>Configurable Extractors</b>	<b>Import Mapping Templates</b>
HP OpenView IT Service Manager 5.6 Migration	Multiple files available, refer to the Service Desk 3.0 Migration Guide.	Multiple import mapping available, refer to the Service Desk 3.0 Migration Guide.

## About Item Mapping

Every external class imported needs to be mapped to an item in Service Desk. Templates exist for every item in Service Desk. Templates provide a way of relating an external class to a Service Desk item and category. Templates contain default attribute values. You can create one or more import mapping templates for each external application using data exchange.

If PC inventory information is being imported you might search for a template for configuration items, selecting a template that comes closest to matching the type of data you will import. There may be a template specifically for PCs available. The template will contain a number of default attributes, and values. You will map the exported properties and values for the PC to those for the configuration item in Service Desk. The mapped data overwrites any information in the default fields of the template when it is imported. Classes and their associated properties and values must be mapped in order to be imported into Service Desk. The class name between [ ] brackets in the configurable extractor file is the class you need to map in the import mapping.

## Attribute Mapping

Attributes define and identify items, for example the exported class PC might have the properties Model and Location, Service Desk may have similar attributes called Type and Address. For those properties to be imported, they must be mapped to attributes that belong to a Service Desk configuration item so that the system knows where to put them when they are imported. ATT and PARENT\_RELATION\_NAME in the configurable extractor determine the attributes that have to be mapped when configuring the import mapping. The term properties has the same meaning as the term attributes used in Service Desk.

Import mapping for a persons gender is done by using 0 for a male and 1 for a female. The attribute name ID is reserved for internal use by the Service Desk application.

## Key Binding

Key binding is used to identify specific external classes, or items in Service Desk. An example is a serial code assigned to a printer, an employee number or even the property PC.Hostname. By setting the key

binding for an attribute you can identify similar items and recognize changes. At least one attribute in every class must be used for key binding, the `Name` property is set as a default key value in the import mapping examples. Select the `This field is used for key binding` check box in the `Field Mapping` dialog box to make that attribute a unique key.

### **Value Mapping**

Many attributes come with values that further define them. Some attributes contain values in a code table, meaning that there is a pre-defined set of possible values to choose from. Other attributes have values that cannot be predefined, such as serial numbers or employee names. When you map an attribute that has values from a code table, the value mapping button will become available in the `Field Mapping` dialog box. You will then be able to map the external property values to the attribute values that exist in the code table in `Service Desk`.

### **Required Fields**

When importing items into `Service Desk` keep in mind that required fields might exist for that item, for example the `Name` field might be required if you are importing a personnel record. If you try to import an item without including the information for the required fields. To view required fields for each item; from the `Tools` menu click `System`, and then expand the `Security` directory from within the `Administrator Console`. Click `Prevention` and then `Required Fields`. More detailed information about using that dialog box can be found in the online help.

### **Defining Relations**

Items are often related to other items. For example, a PC may come with a CD-ROM and have a monitor, a mouse and a keyboard connected to it. It probably has a number of software programs running on it and is connected to a network with a specific IP address. The relation between these items is hierarchical with the PC as the parent, and the components being related to it the children. In a database this structure is stored in the form of an object tree, with the components forming branches or nodes in the tree. The nodes in the tree are named according to their position. The node above another node is a parent while the node immediately under it is the child. The node at the top of the object tree is called the root and does not have a parent. In this example the PC or item is the root of the object tree.

## The Import Mapping

### About Item Mapping

How relations are imported into Service Desk is determined by how they were extracted and how the import mapping is done. Import mapping for relations is done by using default search attributes from Service Desk that belong to each entity. Ensure that you use search codes that exist in Service Desk. The extractor is designed to convert information from the external data source in a consistent way, maintaining the object tree structure. The API matches parent and child relationships of items being imported with that of the items already present in Service Desk and imports the data.

For additional information about configuring the extractor to export relations, see “Defining Relations in the Configuration File” on page 41. For an example see “Importing Data With Relations” on page 193.

The Data Dictionary provided with Service Desk provides a description for all items and tables in Service Desk to include the relations between those items. For example, if the attribute for an item has a 1:1 relation, e.g. Service call caller to the Person item, the table where the other item is stored will be listed. If the attribute has a 1:N relation, for example Service call History lines to History line items, the table where the items are stored will be shown along with the table column that contains the foreign key.

There are three primary types of relations that can be imported; N:1 relations based on a search key, N:N or Parent-Child relations, and Rel:Rel that involve multiple classes. The following sections will explain the relation types in more detail and include information on what you can do in the import mapping to import them correctly.

---

#### CAUTION

Old objects and relations are not automatically deleted in Service Desk. You must manually locate and delete old objects and relations to remove them.

---

#### Search Key; N:1 Relations

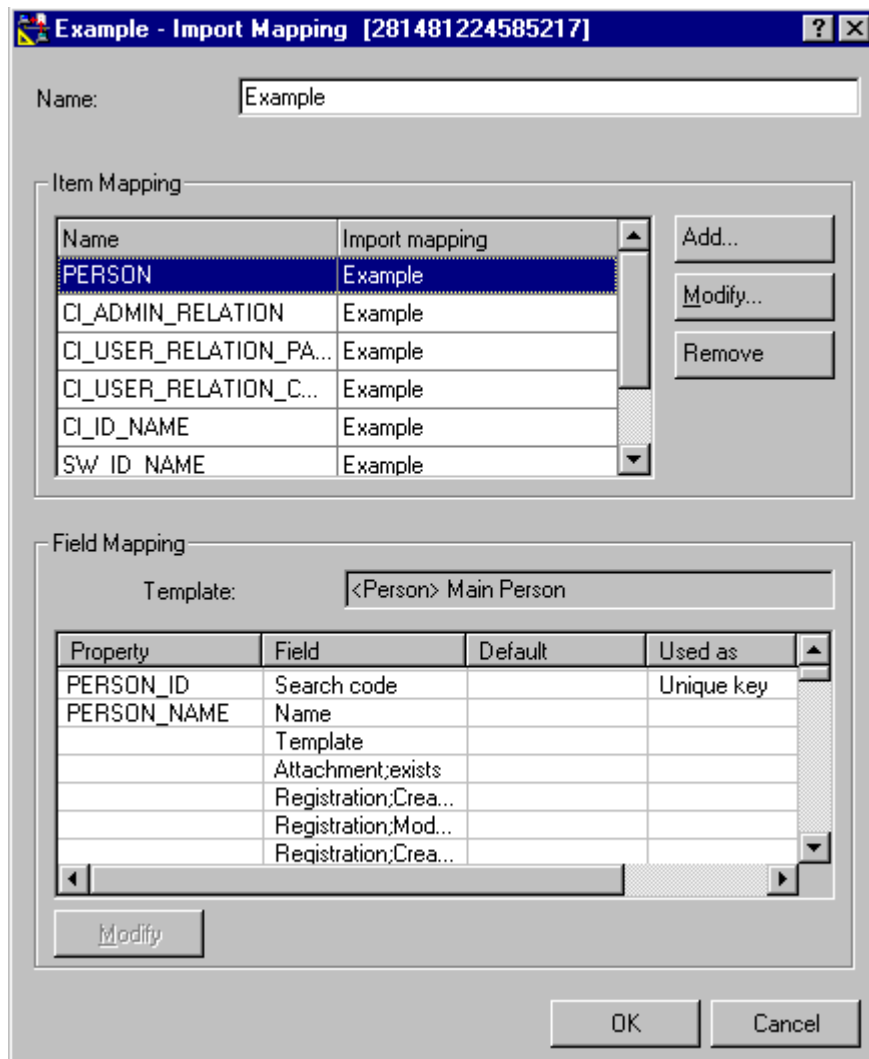
In the N:1 relation the N refers to one or more items in Service Desk related to one (1) other item in Service Desk. This type of relation uses a search code as an entity reference to determine the relationship between the items. The item that is referenced using a search code should be listed first in the exported XML file, the order in the XML file is the order it will be imported in. The order is determined by how the classes are listed in the [CLASSES] section in the configuration file.



See “Search Key; N:1 Relations” on page 64 for a description of how to configure the configurable ini file for this example.

To complete the process for importing this relation you will need to create the import mapping to use a search code as a reference to the item. For example:

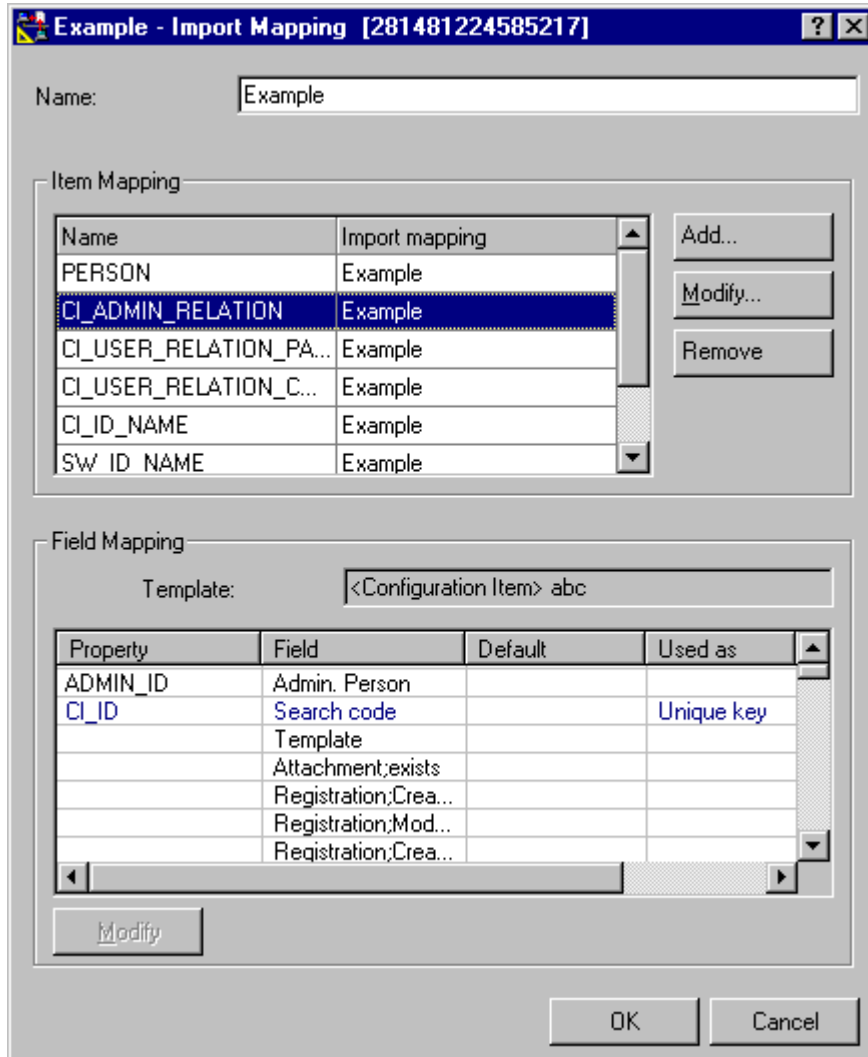
**Figure 3-1 Import Mapping for PERSON Class**



The Import Mapping  
About Item Mapping

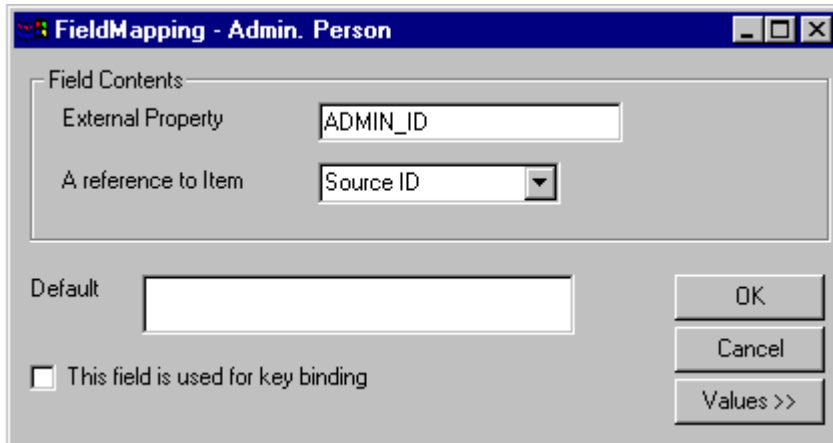
Import Mapping then needs to be done for the CI\_ADMIN\_RELATION Class:

**Figure 3-2** Import Mapping for CI\_ADMIN\_RELATION Class



Next you need to make a reference from one item to the other as is done in the following Field Mapping dialog box:

**Figure 3-3**                    **Field Mapping for ADMIN\_ID**



For an additional example see “Importing Data With Relations” on page 193.

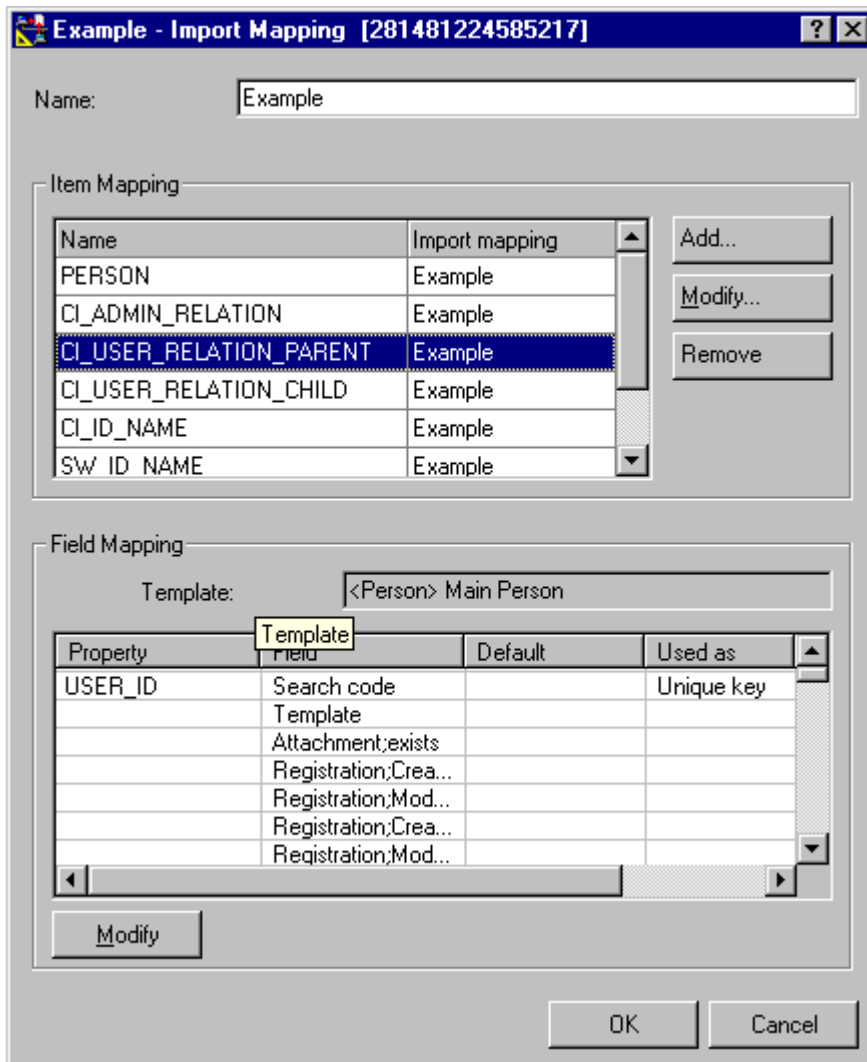
### **Parent-Child; N:N Relations**

In this type of relation the N refers to one or more items in Service Desk related to N, one or more other items in Service Desk. This type of relation uses multiple item search codes to make the relationship. The configuration file is modified differently for a N:N relation than a 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.

See “Parent-Child; N:N Relations” on page 67 for a description of how to set up the configurable ini file for the following example.

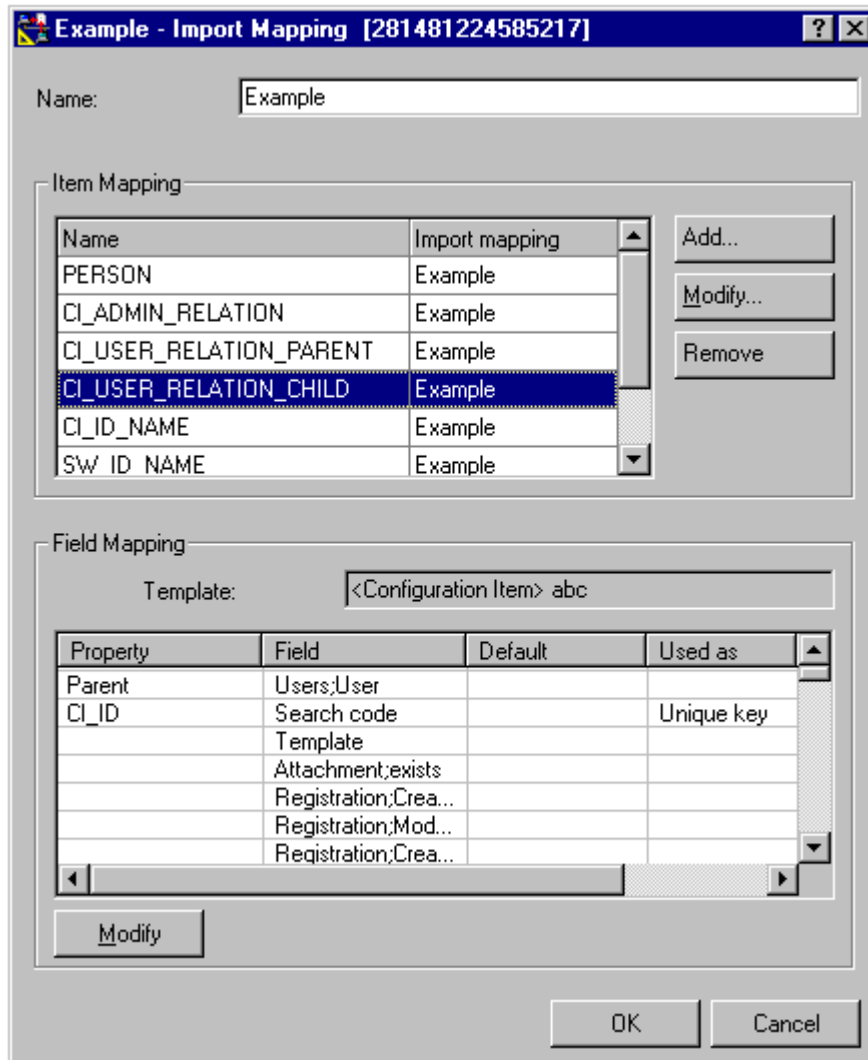
To import this relation you will need to create entity references in the import mapping as follows:

**Figure 3-4** Import Mapping for CI\_USER\_RELATION\_PARENT Class



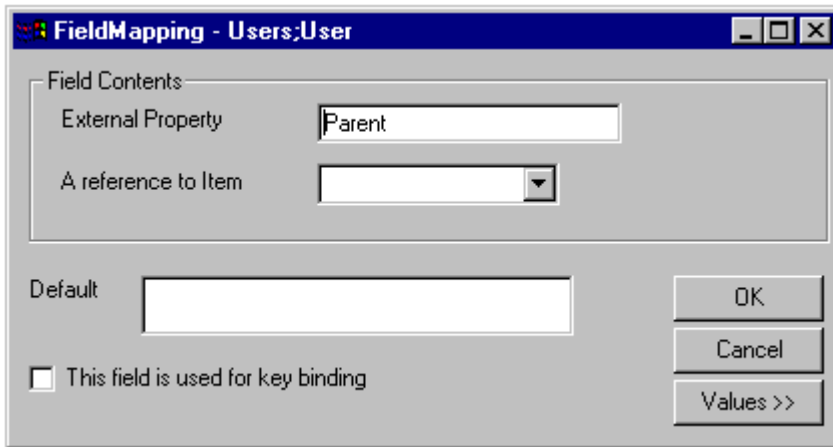
Next you will need to create the import mapping for the child relation:

**Figure 3-5 Import Mapping for CI\_USER\_RELATION\_CHILD**



Next you will need to complete the mapping for the parent-child relation as in the following Field Mapping dialog box:

**Figure 3-6**      **Field Mapping for Parent**



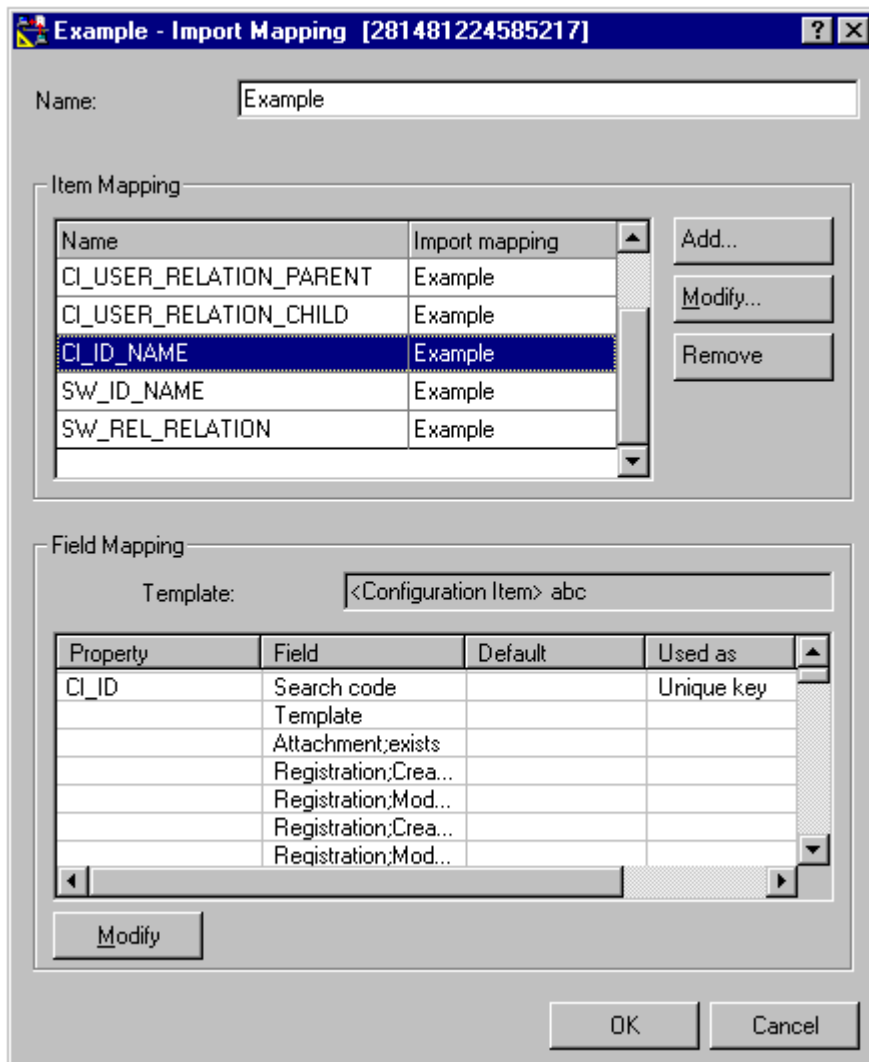
### **Relations with Relation Type; 1:Rel:1 Relations**

The 1:Rel:1 type relation is a relation that is specified by creating another class for the purpose of describing the relation between two or more classes. You can also use an entity reference with a search code and an entity reference to a relation table in Service Desk to describe the relation. When using a reference to a relation table, the relation is maintained in a separate relation code table. You can map items to the values in the relation table to define the type of relation. Another option is to enter a default value or create value mapping for the relations.

See “Relations with Relation Type; 1:Rel:1 Relations” on page 70 for a description of how to configure the configurable ini file for the following import mapping example.

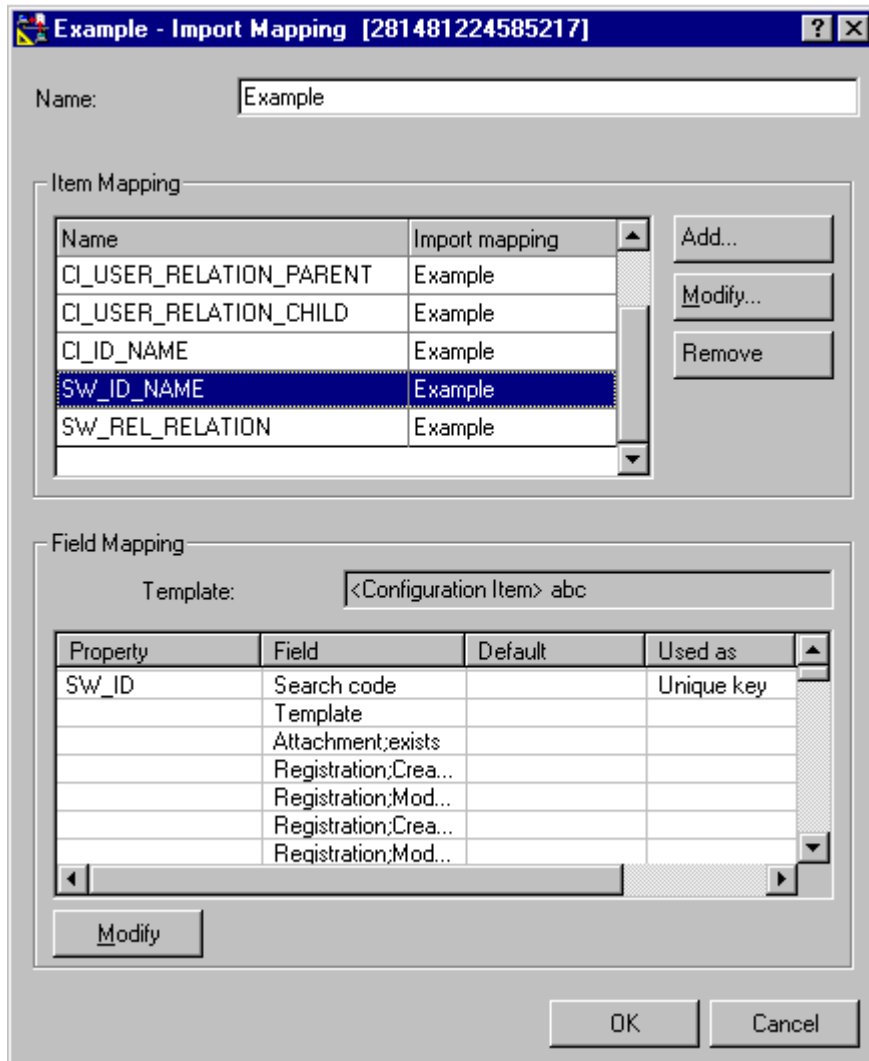
The following dialog boxes show an example of the import mapping for this type of relation. This dialog box shown the import mapping for one class:

**Figure 3-7 Import Mapping for CI\_ID\_NAME Class**



The import mapping for another class is shown in the following dialog box:

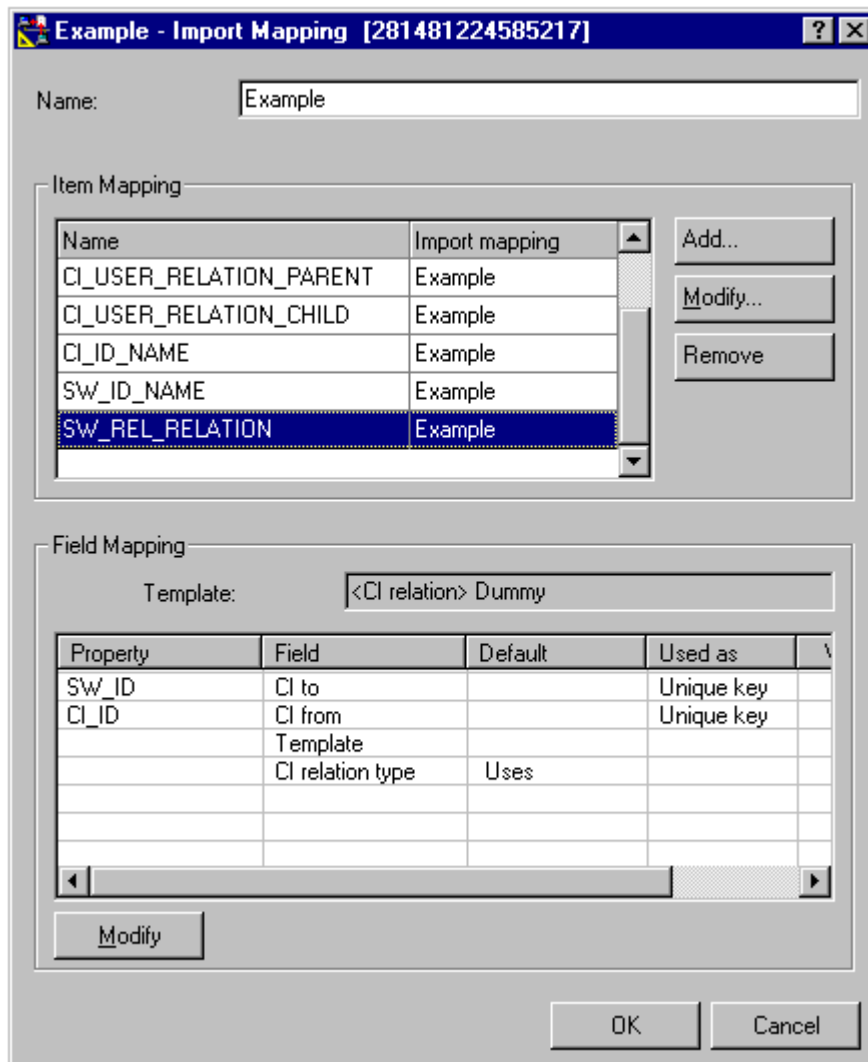
**Figure 3-8**      **Import Mapping for SW\_ID\_NAME Class**



The following class is imported to define the relation :

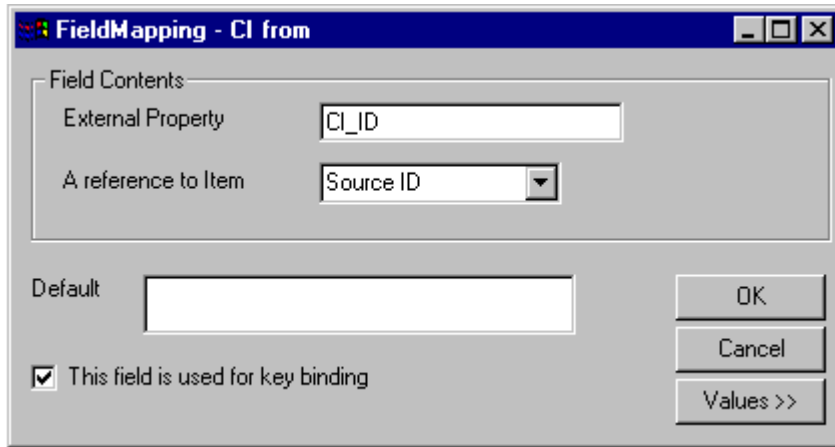


**Figure 3-9 Import Mapping for SW\_REL\_RELATION**

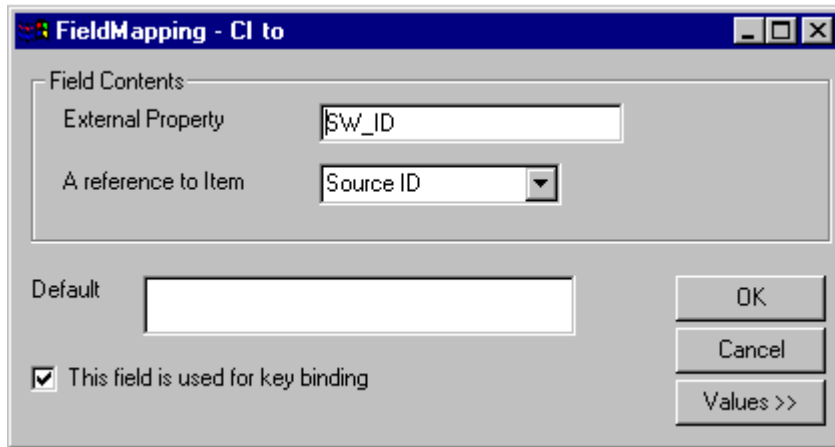


Two entity set references are made to link the classes as shown in the following dialog boxes:

**Figure 3-10**      **Field Mapping for CL\_ID**



**Figure 3-11**      **Field Mapping for SW\_ID**



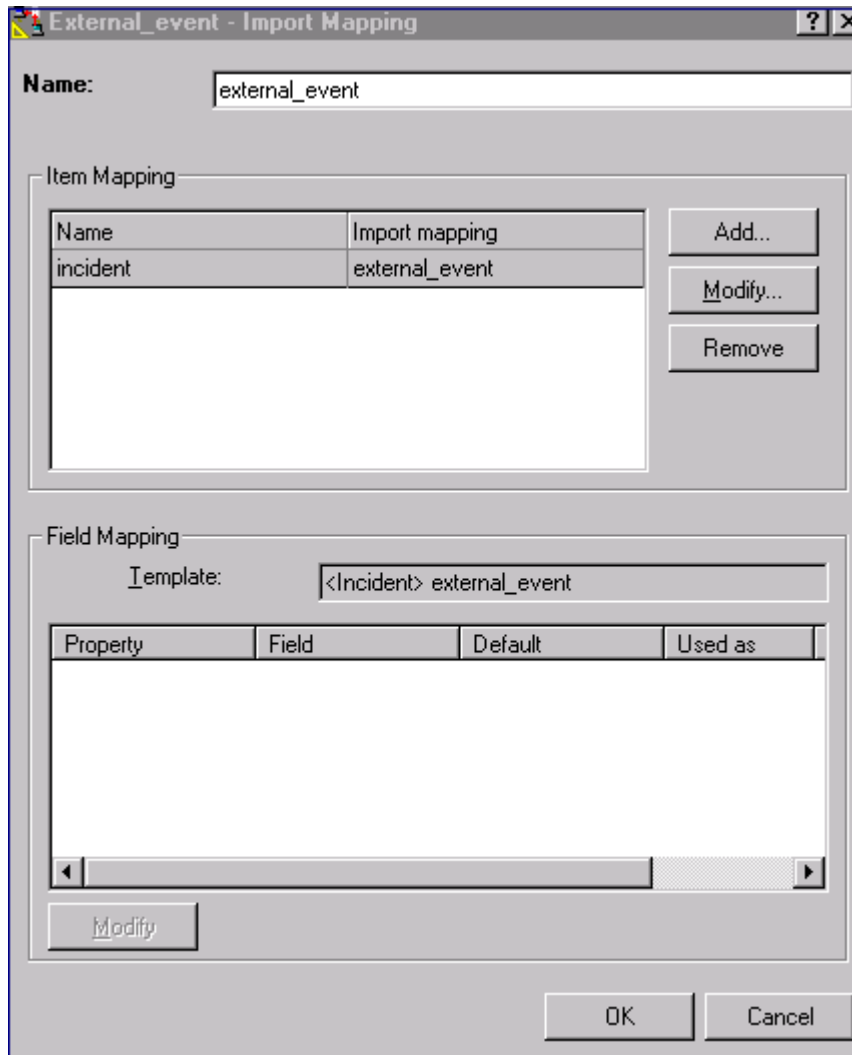
## Mapping Service Events

The same import mapping in Service Desk is used for service events. You can use import mapping settings created for importing batches of data, as long as the class you are importing and the import mapping name are the same. Another option is to set the import mapping especially for the service event you will be importing.

The import mapping `Name` field(-x switch) and the `Class` name (-c switch) must be included in the command line for the service event. For more information see “Importing Service Events” on page 99.

The Import Mapping  
Mapping Service Events

**Figure 3-12** Import Mapping dialog box for sd\_event



## Mapping Classes and Attributes

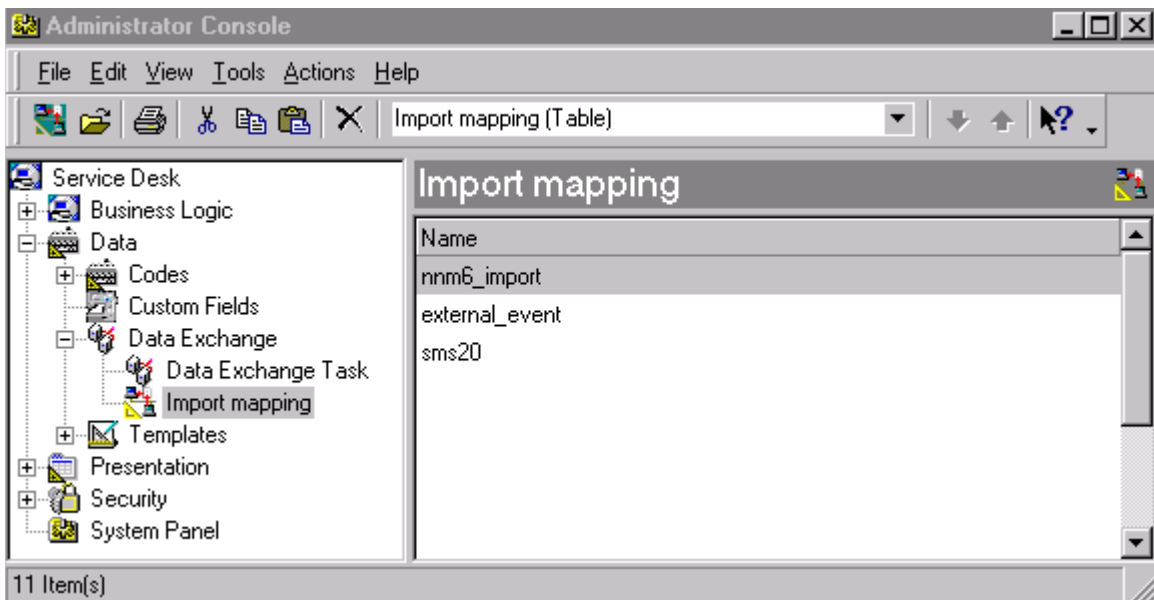
Import mapping can be done from the application server or a client computer. You can create a new import mapping or use the examples we provide. The available examples will be visible in the import mapping window.

### Creating a New Import Mapping

**Step 1.** Navigate to the Import mapping folder:

1. From the Tools menu, click System, then Data, then Data Exchange, and then Import Mapping:

**Figure 3-13** Import Mapping Task Window



**Step 2.** To open a New Import Mapping dialog box, right-click anywhere in the import mapping window. Select New Import Mapping from the menu that appears.

**Step 3.** Provide a name for the import mapping to identify which external

## The Import Mapping Mapping Classes and Attributes

application it is used for. It is recommended that you use a similar name for the import mapping, configuration file, and integration account.

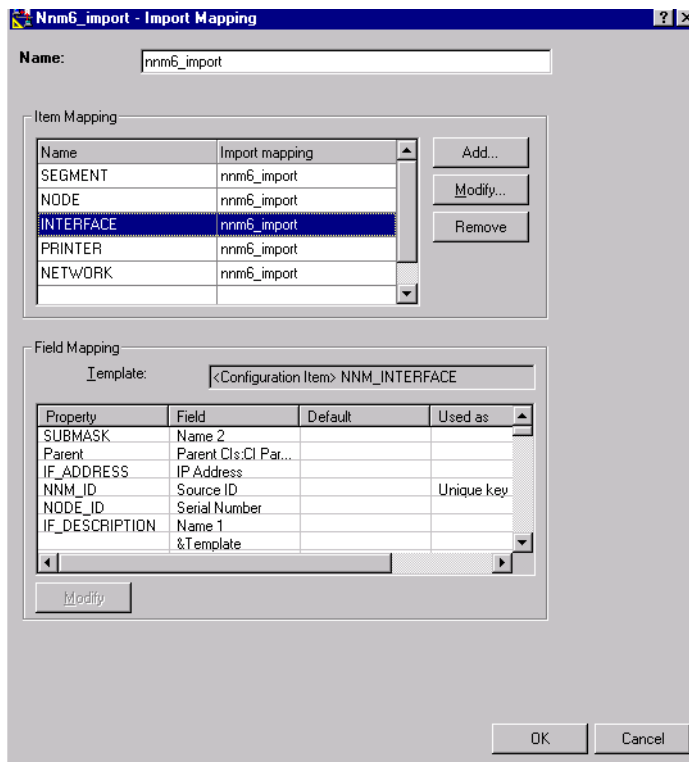
- Step 4.** Map classes, properties and values to Service Desk items, attributes and values. To continue with the mapping process follow the instructions in Step 2, “To add or change a class mapping:” on page 79.

### Modifying an Import Mapping

- Step 1.** Open an existing import mapping:

1. Double-click the `Data Exchange` folder and then double-click `Import Mapping`. All import mappings will be displayed in the `Import mapping` window. Double-click one of the import mappings and the `Import Mapping` dialog box will appear:

**Figure 3-14** Import Mapping dialog box



2. In the Item Mapping portion of the dialog box select the class you want to modify and click `Modify` to change the class selected.

The `Field Mapping` portion of the dialog box will show the attribute mapping for that item:

- The `Property` column shows external properties as they should appear in the XML file.
- The `Field` column shows Service Desk attributes.
- The `Default` column shows the default values from the template the class is mapped to.

---

**NOTE**

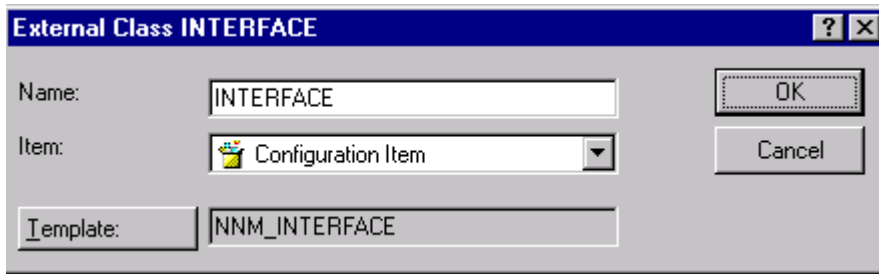
Only editable fields will be shown in the import mapping dialog box. Read-only fields will not be displayed.

---

**Step 2.** To add or change a class mapping:

1. To add a class to your import mapping click `Add` in the Import Mapping dialog box. The `New External Class` dialog box will appear, skip to bullet 3 for instructions on what to enter in the fields.
2. To modify a mapped class, select the class in the Item Mapping window and click `Modify`. This will open the class in the `External Class` dialog box.
3. In the `External Class` dialog box, the `Name` field must contain the name of the external class you want to import. The class name must be entered exactly as it appears in the XML file.
4. Use the drop-down arrow in the `Item` field to select the item you want to map the external class to in Service Desk, this step helps in categorizing the class and narrows the number of templates you will need to choose from:

**Figure 3-15 External Class dialog box**

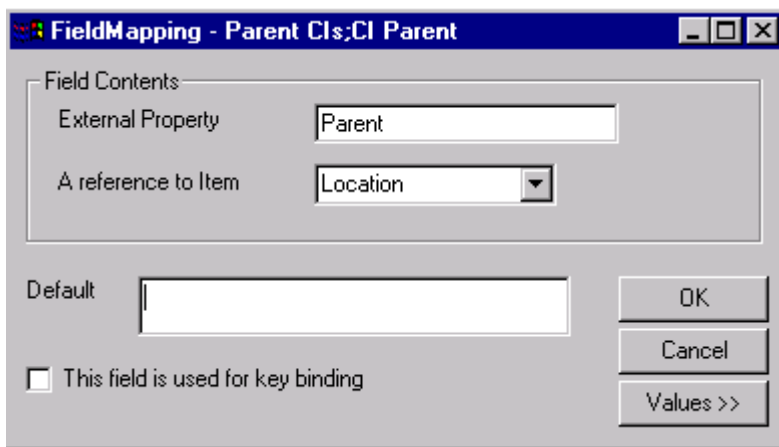


5. Click the `Template` button to select a template from a list of available templates. Every imported class must be mapped to a template. The template will provide default values for the imported class.
6. Click `OK` when done to save your mapping. Repeat this step for each class mapping you want to modify or add.

**Step 3. Modify mapped attributes:**

1. From the main Import Mapping dialog box, select the class the attributes belong to and then double-click the attribute in the `Field` column to map an external property to it. The Field Mapping dialog box will appear:

**Figure 3-16 Field Mapping dialog box**



2. In the `External Property` field, enter the name of the external

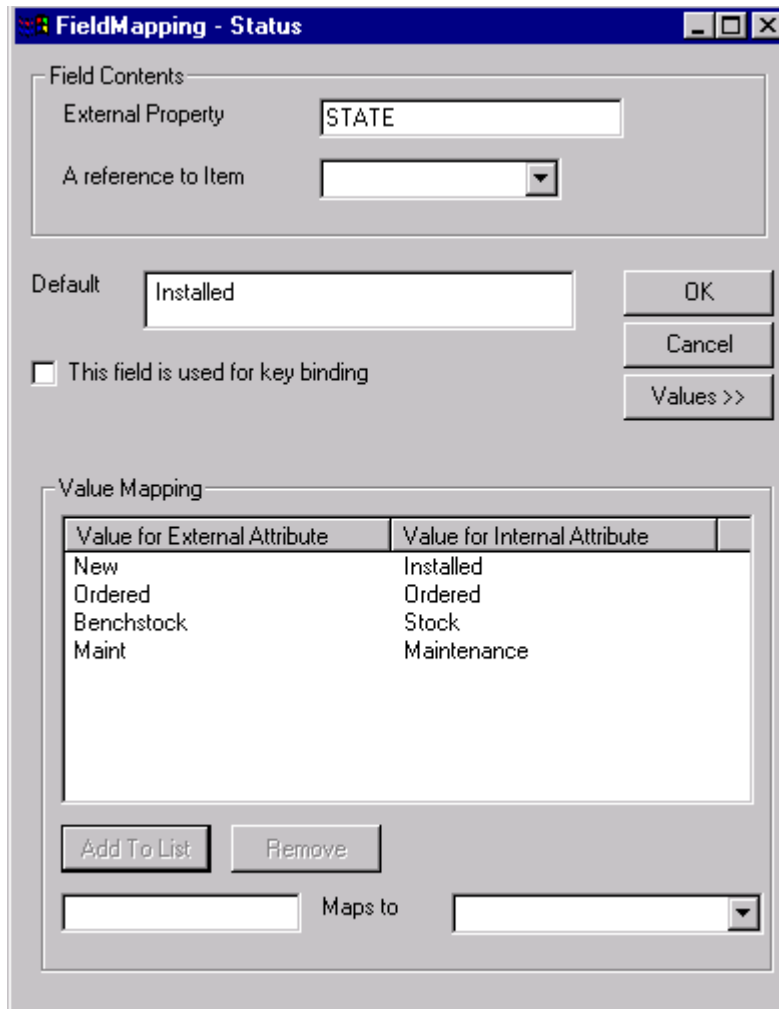


- property you want to map to a Service Desk attribute.
3. You can use the `A` reference to `Item` field to create an additional reference to another item. For example, you can map the `External` property `OWNER` to the attribute `Person` and then reference additional attributes associated with `Person`, for example `address` or `email`.
  4. In the `Default` field you can enter a default value for the attribute you are mapping. This will show up in the `Default` column in the main Import Mapping dialog box. If a property does not contain a mapped value the default will be used in Service Desk.
  5. Select the `This field is used for key binding` check box if the external property contains a unique key value. A class must have at least one property marked for key binding.
  6. Click `OK` when you are finished. If a value list is available for the attribute the `Value` button will be activated, see the following step for more information on mapping values.

**Step 4.** Map property values to attribute values:

1. If a value list exists for an attribute selected in the `Field` column, the `Value` button will become active when you are in the `Field Mapping` dialog box. Click `Value` and the `Field mapping` dialog box will be extended, to include value mapping:

**Figure 3-17 Expanded Field Mapping dialog box**



2. In the bottom left field, enter the value from the external property that you want to map.
3. In the bottom right field, enter the Service Desk value. A drop-down arrow is available for searching for available values.
4. Use the Add To List button after mapping a value. When you have mapped all of the values for the attribute, click OK to return to the External Class dialog box.

5. When you are done import mapping, select **OK** in the Import Mapping dialog box to save your work. If you do not perform this step your mapping will *not* be saved.

---

**CAUTION**

If you change the import mapping of a class that was already imported into Service Desk, and then import that class to a different location in Service Desk, your data may *not* be imported correctly.

---

The Import Mapping  
**Mapping Classes and Attributes**

---

# 4 **Importing Data in Batches**

Service Desk provides you with the flexibility of importing data in batches or importing service events via a command line. This chapter explains the batch import process, chapter 5 “Importing Service Events” on page 99 explains the process for importing service events.

You need to use a configurable extractor for exporting, and an import mapping for importing. A number of example configuration files and import mapping files can be accessed from the Data Exchange dialog box after installing the Integrations tools and the demo database from the Service Desk CD-ROM.

The data exchange software will organize the exported data into a file for importing. The import mapping is applied and the data is arranged so that it can be quickly imported into the database. If data exists that is not mapped, that data will not be imported. The data file is then sent through the server designated for data exchange and the Service Desk database is updated.

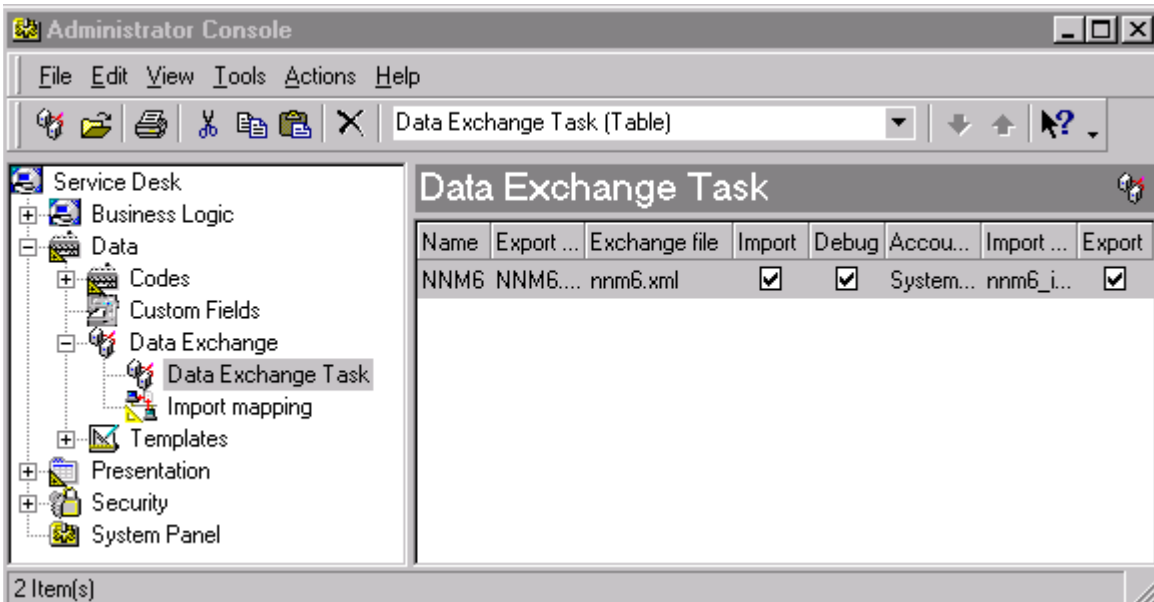
## Using a Task to Import Data

The data exchange process for batch importing is run from the Data Exchange dialog box in the Administrator Console.

**Step 1.** Open a data exchange task.

1. From the **Tools** menu, click **System**, then expand the **Data** folder from within the Administrator Console. Double-click the **Data Exchange** folder, then **Data Exchange Tasks**.
2. Double-click the **Data Exchange Tasks** icon. Existing data exchange tasks will be visible in the window.

**Figure 4-1** Data Exchange Task dialog box

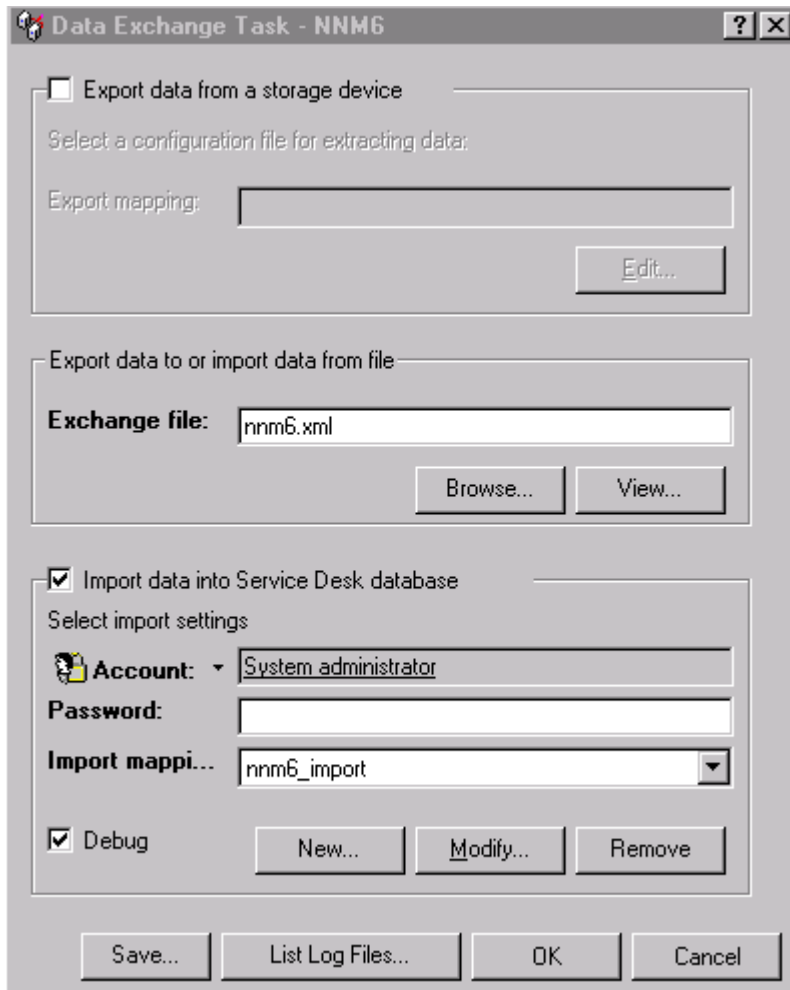


3. Double-click the task you want to use to import data. The Data Exchange dialog box will open.

**Step 2.** Configure the task to import data.

1. Enter the name of the XML file you want to import in the **Exchange file** field:

**Figure 4-2 Data Exchange dialog box - Importing Data**



2. Select the Import data into the Service Desk database check box.
3. Enter the Account established for importing data for this integration and the password for that account. Use the Account drop-down arrow to search for the correct account. For more information see “Integration Accounts” on page 95.
4. In the Import Mapping field use the drop-down arrow to find the



import mapping you want to use during this data exchange. See “Mapping Classes and Attributes” on page 77 for more information about import mapping.

5. Select the `Debug` check box to create a detailed log file while importing.

**Step 3.** Import the data.

1. Click `List log files` to get a list of the log files available for viewing. To select a file for viewing, right-click on the file and the select `Open`. Log files can be opened at any time during the data exchange process.
2. Click `Save` to save the modified task, then `OK` to run the data exchange processes selected. When the export process is complete the sentence “EXTRACTOR finished” will appear in the log file. When the import process is finished the line “Finished loading relations at...” will be shown in the log file.

## Creating Data Exchange Tasks

If you frequently export data using a specific configuration and import mapping, you can shorten the number of steps you must perform by saving those settings as a task. Once saved you will only need to open the task from the Data Exchange folder to run it, instead of entering information in the Data Exchange dialog box again. You can create multiple tasks for exporting, importing, or both exporting and importing data. To create a task:

1. From the **Tools** menu, click **System** and then double-click the **Data** icon in the **Administrator Console**.
2. Double-click the **Data Exchange Tasks** icon
3. Right-click in the **Data Exchange Tasks** dialog box and click **New Data Exchange Task** from the menu that appears.
4. Fill in the fields in the **New-Data Exchange Task** dialog box. For additional information on filling in these fields see “Exporting Data” on page 51 or “Using a Task to Import Data” on page 87.
5. Click **Save** to save the task and **OK** if you want to run the task. The task will be visible in the **Data Exchange Task** window with the options you have set.

---

### NOTE

You can only configure tasks from the application server and *not* from a client computer.

---

## Executing a Task

After creating and saving Data Exchange tasks you can quickly execute them. To execute an existing task:

1. From the **Tools** menu, click **System** and then double-click the **Data** icon in the **Administrator Console**.
2. Click **Data Exchange Tasks**.
3. Double-click a task in the **Data Exchange Task** window to open it.
4. If you agree with the information entered for the task, click **OK** at the

bottom of the Data Exchange dialog box to execute it.

## Scheduling a Data Exchange Task

Data Exchange tasks can be scheduled by using the NT scheduler available with Windows NT. The Schedule service must be running when you want the task performed. Insert the `at` command into the command line used for the data exchange task. For example, the `at` command can be added to the command line syntax for exporting data as follows:

```
sd_exchange export <configfile> <exportlog> <xml file>
at[\\<computername><time>[/interactive][/every:<date>[,...]]
/next:<date>[,...]]
```

The following table explains the parameters used with the `at` command example shown above. For more information refer to the online help for Windows NT:

Parameter	Comment
\\computername	Use to specify a remote computer. Leave out if you want to schedule the command on a local computer.
time	Use to specify the time when the command should be run. Enter the time in 24 hour clock format, for example 22:30.
/interactive	Allows the scheduled process to use the desktop of the user who is logged on.
/every:date[,...]	Enter the day of the week (M,T,W,Th, F,S,SU), or one or more days in the month (1-31). Command will be run every time the day specified occurs.
/next/date[,...]	Same as above except the command will only be run on the next occurrence of the day specified.

---

## Command Line Parameters

Data exchange processes use the `sd_exchange.cmd` file in the Data Exchange folder. When using the Data Exchange features directly from the Administrator Console in Service Desk the application takes care of accessing the commands to export, import, or both export and import combined. To perform these functions from the command line you will need to use one of the following command lines:

**Table 4-1 Complete Importing and Exporting Commands**

Mode	Syntax
Export	<code>sd_exchange export configfile exportlog xmlfile</code>
Import	<code>sd_exchange import inputfile username password mapping debug importlog tempdir</code>
Export/ Import	<code>sd_exchange export_import configfile exportlog in-outputfile username password mapping debug importlog tempdir</code> . You will only need to specify the location of the XML file one time when using this option.

The following sections further explains the commands listed above.

### Parameters for Exporting

The command line `sd_exchange export configfile exportlog xmlfile` can be used from the command line to export data. For example: `sd_exchange export ito.ini ito.log ito.xml`. The following table explains the parameters available:

**Table 4-2 Exporting Command Syntax Defined**

Syntax	Explanation
Configfile	ini file containing the export configuration settings for the <code>sd_export.exe</code> command.
exportlog	The export log file is produced by <code>sd_export.exe</code> .

**Table 4-2 Exporting Command Syntax Defined**

Syntax	Explanation
Xmlfile	This is the name of the output file that will be produced by <code>sd_export.exe</code> .

### Parameters for Importing

The command line `sd_exchange import inputfile username password mapping debug importlog tempdir` can be used from the command line to export data. For example: `sd_exchange import ito.xml ITO_server1 ITO_server1 ito_mapping Y ito.log C:\temp`. The following table explains the parameters:

**Table 4-3 Importing Command Syntax Defined**

Syntax	Explanation
Inputfile	The XML file that will be imported. Should match Xmlfile name.
Username	A Service Desk account used for the integration.
Password	The password used for the account mentioned above.
Mapping	The name of the import mapping that you have defined in the Service Desk application, using Data Exchange.
Debug	Y will turn on the debug feature and provide you with a detailed log. N will turn it off.
Importlog	The log file produced by <code>sd_import.exe</code> .
Tempdir	A folder for placing temporary files.

### Parameters for Importing and Exporting

The command line `sd_exchange export_import configfile exportlog in-outputfile username password mapping debug importlog tempdir` can be used to export and import. You will only need

to specify the location of the XML file one time when using this option. For example: `sd_exchange export_import ito.ini ito_export.log ito.xml ITO_server1 ITO_server1 ito_mapping Y ito_import.log C:\temp`. The following table explains the parameters

**Table 4-4**

**Importing and Exporting Command Syntax Defined**

Syntax	Explanation
Configfile	Configurable ini file containing the export configuration settings for the <code>sd_export.exe</code> command.
exportlog	The export log file is produced by <code>sd_export.exe</code> .
In-outputfile	The XML file, produced by <code>sd_export.exe</code> that will be used to import the data.
Username	A Service Desk account used for the integration
Password	The password that corresponds to the account mentioned above.
Mapping	The name of the import mapping that you have defined in the Service Desk application, using Data Exchange.
Debug	Y will turn on the debug feature and provide you with a detailed log. N will turn it off.
Importlog	The logfile produced by <code>sd_import.exe</code> .
Tempdir	A folder for placing temporary files.

---

## Integration Accounts

All external applications integrating with Service Desk need an account to set authorizations for access to the Service Desk application. Accounts can be created for users who do not need access to the user interface but need to access the application for integration purposes or for using the Self-Service Pages. This type of account can be set from the **Tools** menu. Click **System** and then double-click the **Authorizations** icon from within the **Administrator Console** and open **Accounts**. In the **General** tab select the check box stating: “Use this account for applications and access to the self service pages. This account does not give access to the Service Desk user interface.” In the **account** view you should see a check mark next to your integration in the check box labeled **SSP/integrations** account. Keep in mind the following when creating an integration account:

- You can set a host name for an external application. For example, if you set up an integration to send incidents detected in the network to Service Desk, the Rule Manager can be used to send a confirmation message back to application host that sent the incident. The following example shows how this looks when you send service events to Service Desk from three different ITO servers:

<b>Account</b>	<b>Host name</b>
ito_server1	server1.companyaccount.com
ito_server2	server2.companyaccount.com
ito_server3	server3.companyaccount.com

- Create integration account names that can be easily recognized. Use the same name for your import mapping to avoid confusion. For example:

<b>Account</b>	<b>Import mapping</b>
nnm6_import	nnm6_import
DTA_import	DTA_import

Importing Data in Batches  
Integration Accounts

- Establish all roles that will be needed for the integration to perform its functions. If importing data, access needs to be granted to view and change certain areas within Service Desk, depending on what type of data you import.
- When any account is given access to the user interface, that account will be counted as a registered (paid for) user in Service Desk.

The following example is of an account set up to import data from an NNM application.

**Figure 4-3**      **Integration Account**

NNM Version 6 Import - Account

General | Roles | Active Service Desk sessions

**Display name:** NNM version 6 import

**Login name:** nmm6\_import

Password: \*\*\*\*\*

Confirm Password: \*\*\*\*\*

Host:

Blocked

Use this account for applications and access to the self service pages. This account does not give access to the Service Desk user interface

OK Cancel



---

**NOTE**

Account information to export data from an external ODBC source is set in the configurable extractor file. Service Desk accounts only provide access to Service Desk for the import process.

---

---

**TIP**

Service events can operate with any valid Service Desk account. Using an account developed especially for the external application makes it easier to determine which external application a service event originated from.

---

Importing Data in Batches  
**Integration Accounts**

---

# 5 **Importing Service Events**

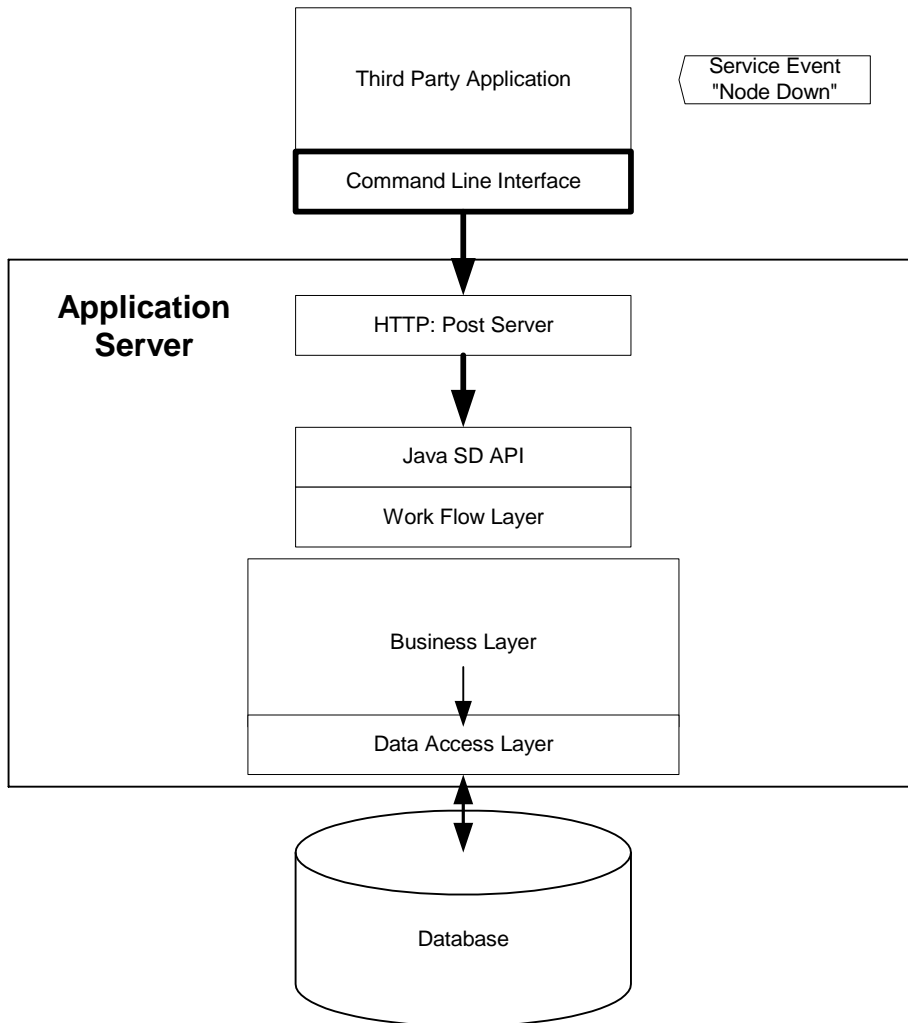
You can use a command line to import individual classes into Service Desk. For example, when a network management application detects a new node, the node information can be automatically inserted into Service Desk by the service event command, `sd_event.exe`.

---

## The Service Event Import Process

To import service events a command line is entered in the external application. The command line directs the service event data into the Service Desk database:

**Figure 5-1**      **Inbound Service Events**



## Service Desk Event Command Line

`Sd_event.exe` makes it possible to insert, update, or delete an item in Service Desk. The command must be entered on the command line as follows:

```
sd_event.exe[-l log file][-s server][-p port][-a
account/password][-x mapping][-c class][-m modus][-v Value list]
```

**Table 5-1**

**Service Event Switches**

Switch	Name	Description
-l <small>(small L)</small>	LOG FILE	Name of the file where information about the server is recorded.
-s	SERVER	Name of the application server used to find the HTTP:Post process
-p	PORT	The port number to access the HTTP:Post process, (default is 30980).
-a	ACCOUNT/PASSWORD	A Service Desk user account, with a forward slash / followed by the account password.
-x	MAPPING	The name of mapping to be applied for default classes, property mapping and value mapping.
-c	Class name	The name of the class defined in the Import Mapping dialog box, for example: pc, node, network, incident.
-m	Modus	Modus being used: insert, update or delete.
-v	Value list	List with property names and property values [property=value]. You must use existing Service Desk values.

The `-x` mapping and the `-c` class name options are taken from the import mapping set in Service Desk. You can use an existing import mapping or create a new one for every service event, as long as the import mapping used includes the class you want to import into Service Desk. For more information about import mapping see “Mapping Service Events” on page 75.

### Special Characters

`sd_event.exe` can handle the use of special characters like tabs, single and double quotes, percent signs, dollar signs and numerous international characters. Since `sd_event.exe` is a command line tool, the following characters need to be represented differently:

New line	<code>\n</code>
Tab	<code>\t</code>
Double quote	<code>\"</code>

---

#### NOTE

`Sd_event` is 8-bit and uses the UNICODE character set. Any additional character code conversion necessary when using other character sets with `sd_event` is the users' responsibility.

---

### Using the Service Event Configuration File

The default name for the configuration file is `.\sd_event.ini`. This file will need to be modified to fit your organizational needs. The following example configuration file is configured to insert incidents from ManageX:

#### Example 5-1

#### Example Service Event Configuration File

```
[SD_EVENT]
LOGFILE=c:\temp\sd_event_managex.log
ERROR_LOGFILE=c:\temp\sd_event_managex_error.log
ACCOUNT=managex4_event/servicedesk
SERVER=localhost
```

```
PORT=30980  
MAPPING=external_event  
CLASSNAME=incident  
MODUS=insert  
LANGUAGE=GB
```

The parameters in the configuration file are used for the following:

**Table 5-2**      **Service Event Configuration File**

Parameters	Definition
LOG FILE	Log file name
ERROR LOGFILE	Error log file name
ACCOUNT	login name/password
SERVER	server name
PORT	port number
MAPPING	mapping name
CLASS NAME	class name
MODUS	modus type (insert, update, delete)
LANGUAGE	Language for error messages

---

**NOTE**

Values that come from parameter switches will take precedence over default values in the configuration file.

---

**TIP**

A quicker method of using the `sd_event` command line is to put parameters in the configuration file and use a reference to the configuration file in the command line with the `-f` switch and a values list `-v`. The structure is:

```
sd_event.exe -f<ConfigurationFileName.ini> -v <Value List>
```

## Importing Service Events

### The Service Event Import Process

for example:

```
C:\project\revelation\bin\sd_event.exe -f sd_event.ini -v  
description="test 1234" status=s1 event_id=654321  
information="my info" impact=2 priority=Low ci=SERVICEDESK
```

---

### **Service Event Examples**

Detailed information and examples for exchanging data using the service event command are available for:

- HP OpenView Network Node Manager, see “Integration With Network Node Manager” on page 131 for detailed information.
- HP OpenView ITO, see “Integration With ITO” on page 145 for detailed information.
- HP OpenView ManageX, “Integration With ManageX” on page 163 for detailed information.



---

## Resending Service Events

When events are sent to Service Desk and Service Desk cannot be reached you don't want those events to be discarded. You can save events that fail to make it to Service Desk in an error log file and send them again. To send the failed events, rename the error log file so that you can continue to gather incoming errors while you send the failed events. Use the command line:

```
sd_event.exe -f [error log file] -r
```

For example, to resend one or more events this will look like: `sd_event -f sd_resend.log -r`. You can add an optional `-b` to resend all of the service events. The import mapping that was used when the error log file was created will be used to resend the service events.

The default location for the error log file is `.\sd_event_error.log`. An example error log file follows:

### Example 5-2 Error Log File

```
[ EVENT_280 ]
VALUE_LIST="username=service#password=desk#mapping=scotttiger#
className=EMP#modus=INSERT#EMPNO=1234#ENAME=Mr.
Jones#DEPTNO=20#"
SERVER=server1
PORT=30980
CLIENT_ERROR=
LANGUAGE=GB
TRY=1
LOGFILE=sd_event_scott.log
ERROR_LOGFILE=sd_event_error_scott.log
TIMESTAMP= 4/14/2000 17:04:27
SEND=true

[ EVENT_371 ]
VALUE_LIST="username=service#password=desk#mapping=scotttiger#
className=EMP#modus=INSERT#EMPNO=1234#ENAME=Mr.
Jones#DEPTNO=20#"
SERVER=server1
PORT=30980
SERVER_RESPONSE=ERROR: You must fill in the Search code box.
LANGUAGE=GB
TRY=1
```

Importing Service Events  
**Resending Service Events**

```
LOGFILE=sd_event_scott.log  
ERROR_LOGFILE=sd_event_error_scott.log  
TIMESTAMP= 4/14/2000 17:04:52  
SEND=true
```

## Preventing Event Storms

When hundreds of events occur at almost the same time it is referred to as an event storm. If unprepared for, event storms can degrade the performance of your system. You can prevent event storms by implementing the tools explained in this section, which will help control how incoming events are handled.

---

### NOTE

Service Desk is not designed to handle a large number of events administratively. Normally it is the task of the event source application to handle event storms with an event correlation system.

---

## Queuing Events

To prevent event storms, you will need to implement the following queuing tools for all your service event processes. The queuing tools manage incoming events so that they are handled one at a time. Three tools and a script are provided for this purpose:

- *Enqueue tool*: responsible for adding new entries to an existing queue.
- *Dequeue tool*: removes the entries one by one and executes the command in each entry.
- *Queuectl tool*: Allows for blocking, pausing, flushing and reporting the status of the queue.
- *Addentry.sh script*: Adds entries to the queue and starts the dequeuer as necessary. (The NT version of this script is called `addentry.cmd`)

### Creating a Queue

A queue is a directory containing a file for each queue entry. You can create a new queue with `#mkdir clock_queue`. An example of this on UNIX:

```
# mkdir clock_queue
# ./enqueue clock_queue xclock
# ./dequeue clock_queue
# for i in 1234
do
```

## Importing Service Events

### Preventing Event Storms

```
./addentry.sh clock_queue xclock  
done
```

If the command for an item to be queued contains any spaces, it must be enclosed by quotes. On UNIX machines, use single quotes (apostrophes) if the command itself contains quotes, and if you use variables. For example: `./enqueue testqueue "my command $USER"`

On Windows NT machines, the quotes that are part of the command must be preceded by a backward slash: `\"`.

For an example see, "The Event Queue" on page 160

### Enqueue

`enqueue <queuname><command>` creates a new entry in the queue containing the command. The queue directory must be created prior to this action. The following exit codes are also applicable for `dequeue` and `queuctl`:

Exit Code	Description
0	The command was queued successfully
1	A command line parameter is missing (usage error).
2	There was an error accessing the queue because it was locked or possibly non-existent.

### Dequeue

`dequeue <queuname>` processes the entries in the queue by reading, executing, and deleting the entries one by one. A new process is started for each entry in the queue. The dequeuing process pauses until each new process finishes. The dequeuer exits when all entries in the queue are executed or when the queue is blocked.

### Queuctl

`queuctl <queuname> <flag>` This tool controls access to the queue

processes. Available flags are:

<b>BLOCK</b>	Blocks all access to the queue. If a dequeuer is running it will exit. No new entries will be added.
<b>UNBLOCK</b>	Removes blocks from the queue. Entries can again be added to the queue, and the dequeuer can be restarted.
<b>ENTRIES</b>	Returns the number of entries in the queue.
<b>STATUS</b>	Returns the current status of the queue. Possible STATUS return codes are:  [IDLE] = exit code 10, no dequeuer is running on this queue.  [RUNNING] = exit code 11, dequeuer is busy procesing this queue.  [BLOCKED] = exit code 12, queue has been blocked by queuctl.  [HUH] = exit code 13, queue is in an unknown state.  Exit code 0= command completed succesfully.

### **Addentry**

`addentry.sh <queuename><flag>`, and `addentry.cmd<queuename><flag>` for Windows NT. Adds entries to a queue, and starts the dequeuer if necessary. The script file calls the `enqueue` tool, and the `queuctl STATUS` and will start the dequeuer if the queue specified is in the STATUS IDLE.

## **Service Desk Event Communicator and Rule Manager**

Bi-directional service event integrations, (eg. the example ManageX and ITO integrations) use the Rule Manager to send event information from Service Desk. Rule Manager agents are used to execute the commands on the third-party application. You will need to install the event communicator, including the `sd_event` program and the Rule Manager agent for the bi-directional integrations. For detailed installation information see the *HP OpenView Service Desk:Installation Guide*.

Uni-directional integrations, that send events to Service Desk , use the `sd_event` program in the Event Communicator and do not use the Rule Manager.

---

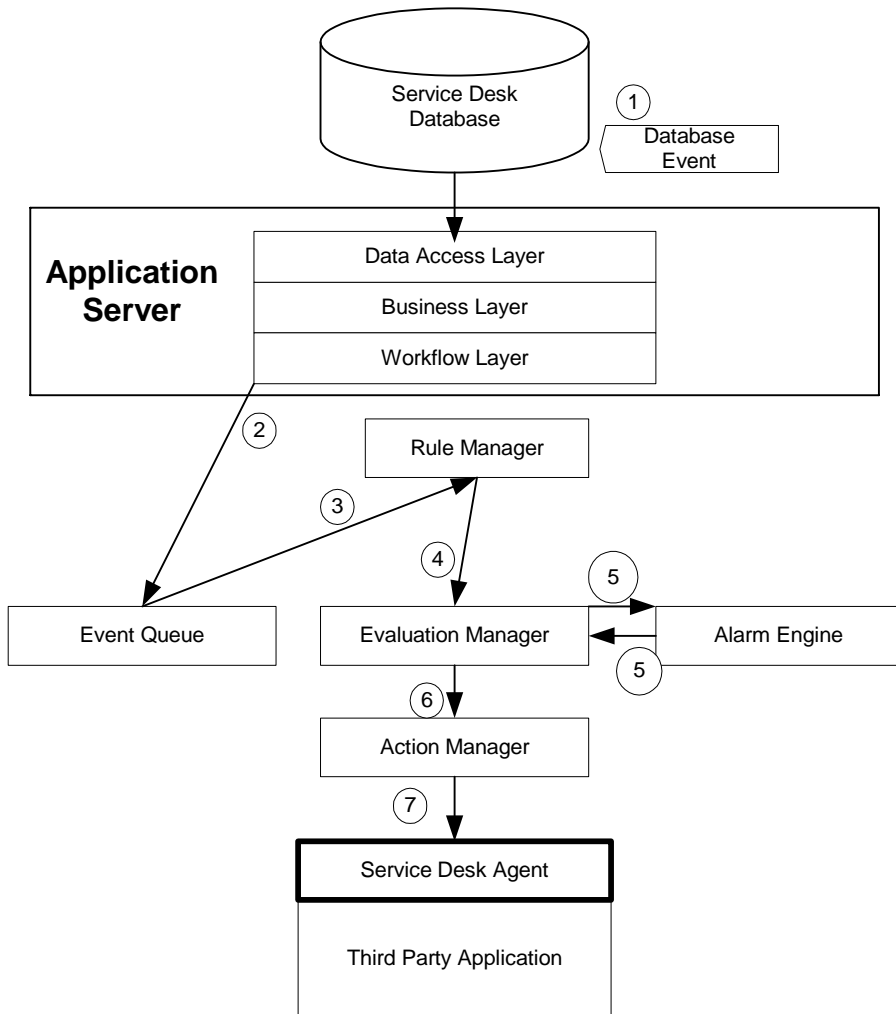
## **6 Outbound Service Events**

Events in the Service Desk database can be sent to an external application through the use of the Rule Manager. An event in the database is defined as an insertion, deletion or modification in a database record. Each event has a date of occurrence, and a session ID.

## The Outbound Service Event Process

The following diagram contains numbers referencing what action is being performed by the application at each numbered step in the process:

**Figure 6-1**      **Outbound Service Event Concept**





1. An event occurs in the Service Desk database.
2. The event is sent to the Event Queue by the Service Desk workflow layer.
3. The Rule Manager takes the event from the queue and checks to see if there is an applicable rule for the event. If a rule does not exist for the event, the event will be discarded.
4. Events that fit a rule are sent to the Evaluation Manager to check the conditions set for the rule.
5. If no conditions exist or if all conditions are met the event is sent to the Action Manager and executed.
  - Events with timed conditions are sent to the Alarm Engine.
  - An alarm goes off at the time specified by the timed condition and the timed condition is evaluated by the Evaluation Manager. Non-timed conditions are not re-evaluated, they are only evaluated once.
6. When the timed condition is met the event is sent to the Action Manager.
7. The Action Manager sends the event to the Rule Manager Agent to be executed.

---

**NOTE**

E-mail actions and Update Service Desk database actions do not use a Rule Manager agent.

---

## The Rule Manager User Interface

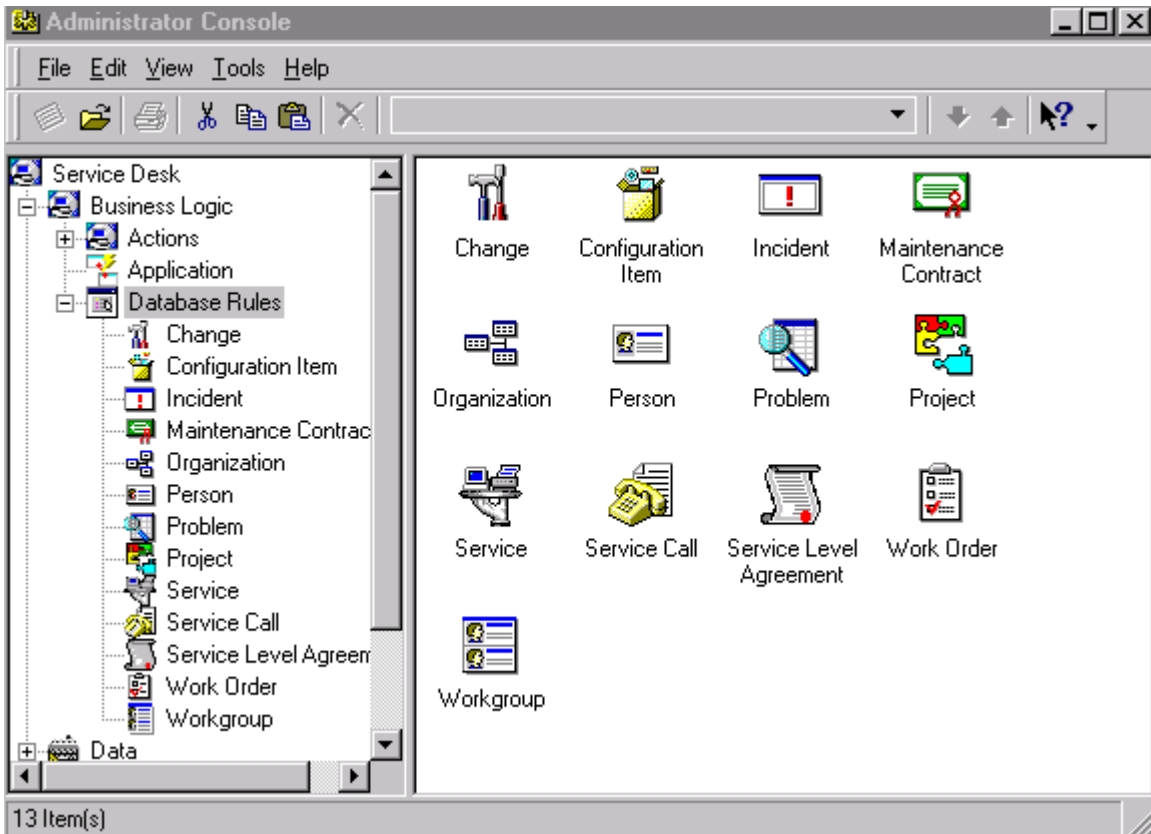
The Rule Manager in Service Desk is the starting point for setting up database rules to send outbound service events. Rules contain the event type, conditions, timed conditions and actions to be performed. Rules are maintained by the Rule Manager and stored in the Service Desk database. The Rule Manager also performs the vital task of pulling event information from the event queue to determine if the rules apply to the event.

### Creating a Database Rule to Send a Service Event

The following procedure is used to create a database rule for sending a service event from Service Desk to another application. The following example shows how a database rule is created for sending a service event back to the ManageX application:

- Step 1.** Select the item in Service Desk that this rule will apply to:
1. From the `Tools` menu select `System`. In the Administrator Console double-click `Business Logic` and then `Database Rules`.
  2. Select the `Incident` item:

**Figure 6-2**      **Select Database Item**

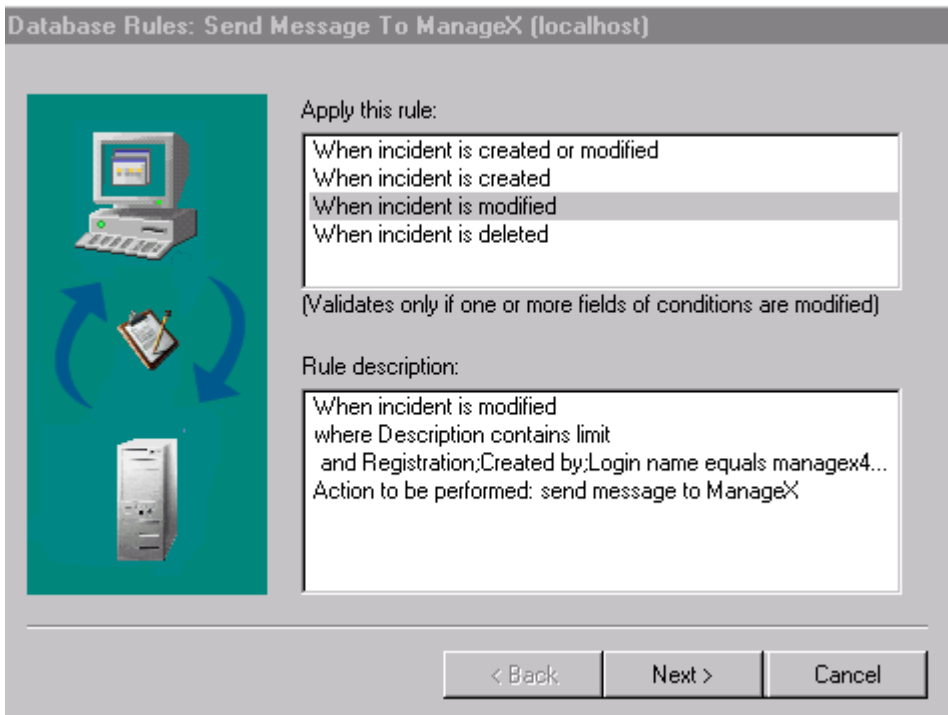


3. Right-click in the Database rule window and select New Database rule from the menu that appears.  
As an alternative you can select one of the example rules we provide and modify it to fit your organizational needs.

**Step 2.** Set conditions for the database rule.

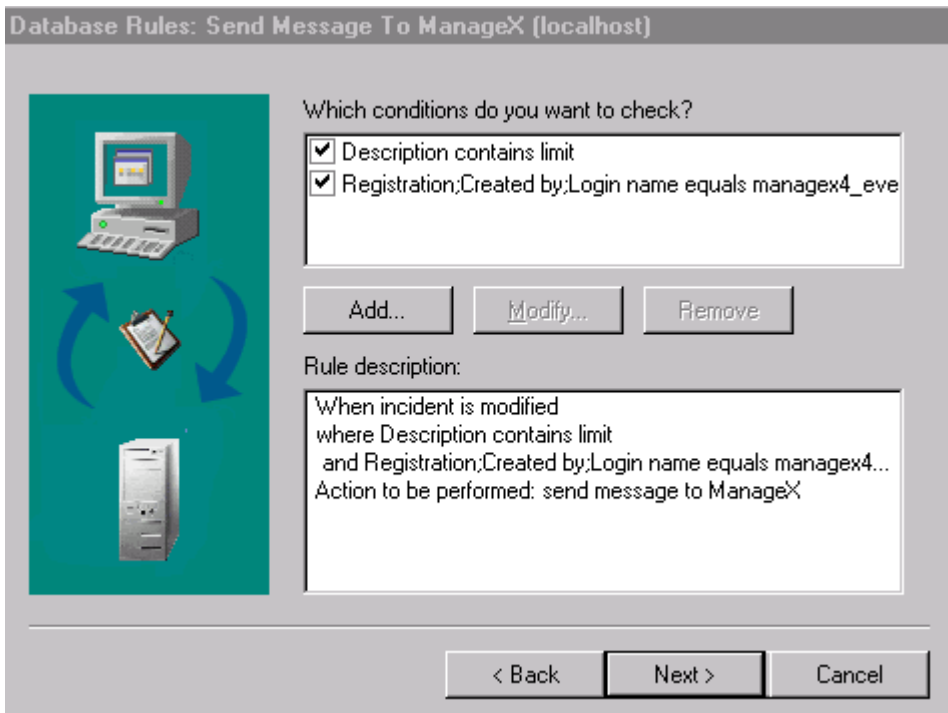
1. Specify when you want to apply the rule. For this example we want to send a service event when an incident is modified, so select “When an incident is modified”. Click Next to continue.

**Figure 6-3 Set Database Condition**



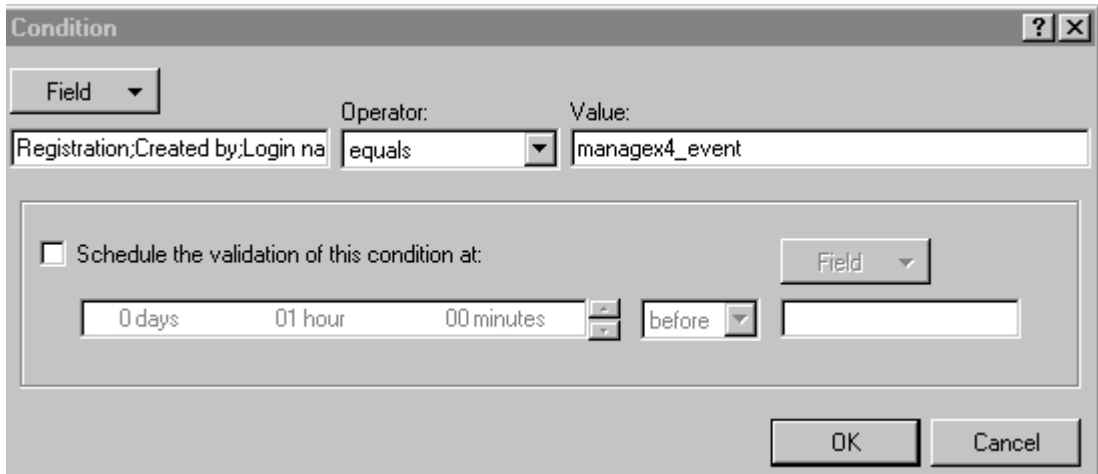
2. In the following dialog box, click Add to set conditions for the database rule:

**Figure 6-4** Add Conditions



3. Use the `Field` button in the `Condition` dialog box to enter the Incident fields you want the rule applied to, you must enter at least one field. Two conditions were entered in this example:

**Figure 6-5**      **Configure Conditions**

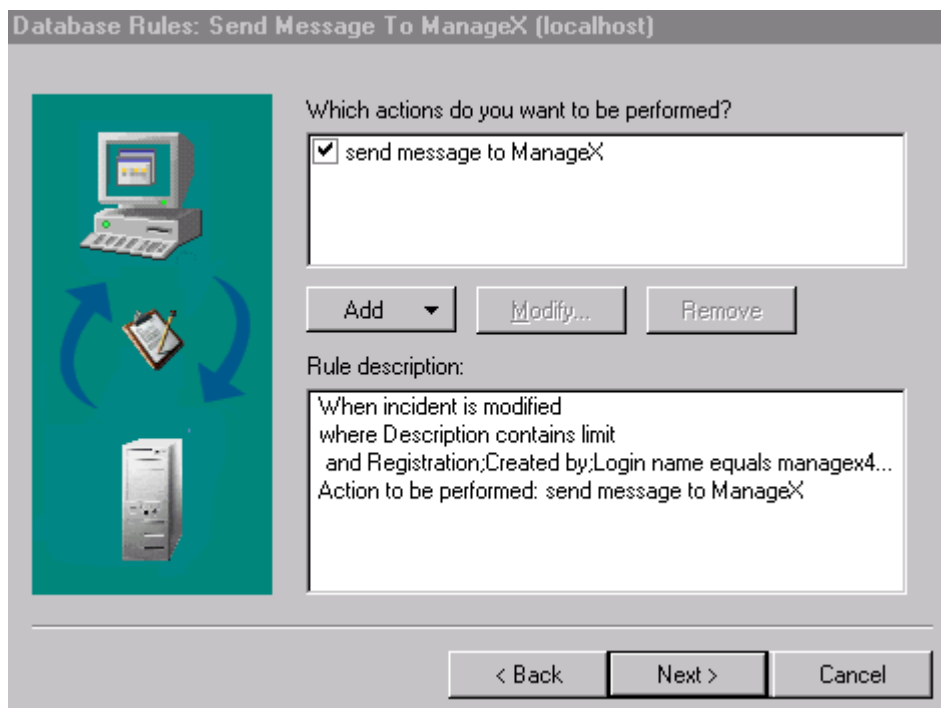


4. Enter an Operator using the drop-down arrow.
5. Put your cursor in Value field and use the Field button to select the value you want the fields selected to be measured by. For example, this rule will be applied when the Incident field Registration contains the value; managex4\_event.
6. Select Schedule the validation of this condition at check box to enter a timed values for this condition. Set a time range, and then a field using the Field button to measure the time value against. This is not a mandatory step.
7. Click OK, then Next to continue.

**Step 3.** Add actions to the database rule.

1. Click Add to add an action. You must specify at least one action for the rule. To send a service event select the Command Exec Action and configure it:

**Figure 6-6** Add Actions



2. Enter an identifying Name for the action:

**Figure 6-7 Create a Command Exec Action**

Command Exec Action

Name: send message to ManageX

Description:  
send message to ManageX

Host  
This command will be executed on the following host:  
host name

Blocked

Command line: sendmessage.exe

Parameters:

Insert at cursor position: Field

OK Cancel

3. In the `Description` field enter a brief explanation of the action that will be performed.
4. In the `Host` field, enter the name of the computer the action will take



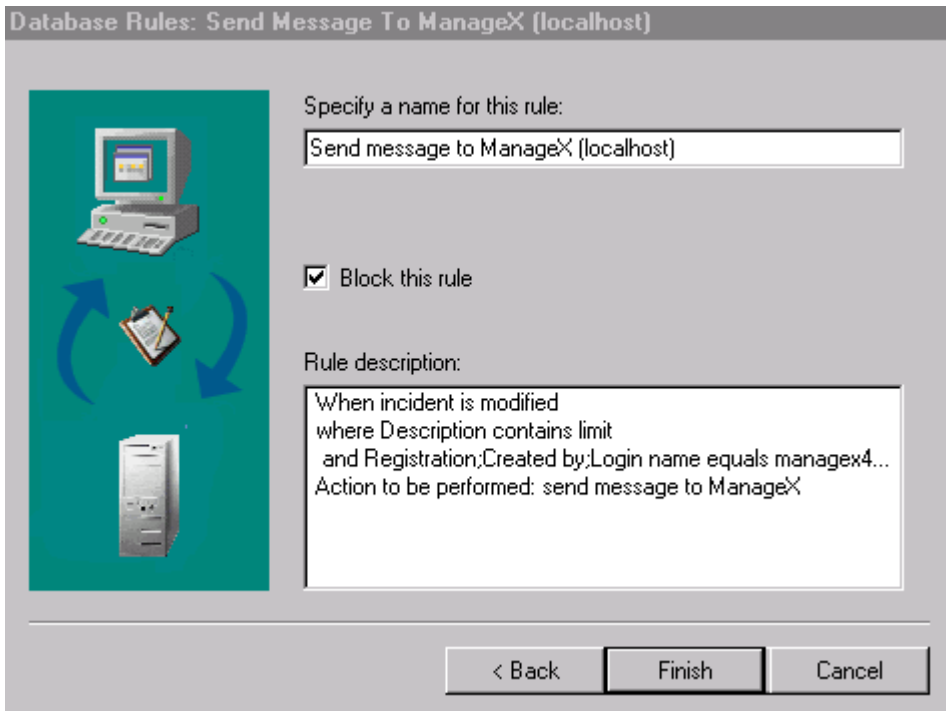
place on. In the `Host` field you can use the `Field` button to specify the `[Registration>created by> hostname]`. If the action is in response to a message sent by an external application the host name can automatically be taken from that account and filled in. For example, if a message is being sent to `ManageX`, the host will come from the applicable Service Desk account, for example `ManageX_event`, that originally sent the incident to Service Desk.

5. Clear the `Blocked` check box to activate this action.
6. In the `Command line` field, enter the command you want to execute, for example: `sendmessage.exe`
7. In the `Parameters` field define additional information for the command by inserting parameters using the `Field` button, for example:  
  
`Message number=[ManageX Message]` (The `ManageX` message number will be inserted when the service event is sent)
8. Click `OK` and then `Next` to continue.

**Step 4.** Turn on the action:

1. In the `Name` field enter an identifying name for the rule. This name will appear in the database rule window for that item.
2. Clear the `Block this rule` check box to turn the rule on.
3. Click `Finish` to save the rule and return to the Administrator Console:

**Figure 6-8 Turn on the Rule**



For additional information about the integration with ManageX, see “Integration With ManageX” on page 163.

## Resending Failed Service Events

When a service event cannot reach the agent it is intended for, that action is stored on the Service Desk application server with the agent’s name. When the agent is active again it calls the Service Desk application server and gathers the actions assigned to it.

---

### NOTE

If your Service Desk application server is shut down, the service events that are stored on the server waiting for an agent to become active will not be saved.

---

---

# 7

## Auditing and Troubleshooting

There are a number of options available for auditing and troubleshooting data exchange. You can use the Viewer to view the exported XML file, you can analyze log files generated during the export and the import process, and you can use the `Debug` mode to create a detailed import log.

## The Viewer, Log Files, and the Debug Mode

You may be able to locate data exchange problems by viewing the exported file with the Viewer. The Viewer will automatically translate your XML file into HTML format and present it in an object tree structure in your browser. The object tree view gives a clear picture of what was exported into the data exchange file. To view an XML file, click `View` in the Data Exchange dialog box.

For more information on the Viewer See “Viewing Data Exchange Files” on page 57.

Another option available when checking the data exchange process is to look at the log files. Log files can be created for both the export and import processes by selecting that option from the Data Exchange dialog box. Log files are located in the `data_exchange\log` folder after they are created.

A detailed log file of the import process can be generated by selecting the `Debug mode` from within the Data Exchange dialog box. The `Debug mode` will generate a very detailed log of the entire import process. It can help you in identifying items that were not imported and potential causes, such as inaccurate mapping, or other errors.

## Troubleshooting

The data exchange process can only be run from the application server or from a data exchange server. If you are importing large quantities of data, a dedicated data exchange server is needed. If the data volume is low, the application server is all that is required. Configuration, XML, and log files can also only be accessed from the server and not from a client machine. The only action you can perform from a client computer is the import mapping process. The following directories must be present in the application server's directory:

- data\_exchange\config
- data\_exchange\XML
- data\_exchange\log

The `sd_export.exe` file, its dll files, and the Windows NT command scripts must be installed in the `bin` folder located in the Service Desk directory. The items in the following table must be in the correct folder for the data exchange process to work correctly:

**Table 7-1 File Directory**

File	Folder
sd_export.exe sd_import.exe sd_import_export.exe	bin
sd_exchange.cmd	bin
msvc60.dll version 6.00.8168.0	\WINNT\SYSTEM32
mfc42.dll version 6.008168.0	\WINNT\SYSTEM32
msvcrt.dll version 6.00.8397.0	\WINNT\SYSTEM32
sd_event.exe	bin
nmm6.ini	data_exchange\config

**Table 7-1 File Directory**

<b>File</b>	<b>Folder</b>
dta5.ini	data_exchange\config
sms20.ini	data_exchange\config
extractor.xsl	repo\xsl

If you receive an error when trying to export, view or import data, one of the following problems may be causing it:

**Table 7-2 Problem Solving**

<b>Problem</b>	<b>Cause</b>	<b>Solution</b>
The XML file cannot be viewed.	The XML format is incorrect.	Check the configuration file and check the log files made during the export process to locate the error.
The XML file cannot be viewed.	The XML file is incomplete	Check the configuration file and check the log files made during the export process to locate the error.
The XML file cannot be viewed.	The DTD file is not in the correct location.	The file CIM_DTD_V20.dtd must be in the same file as the XML file.
The XML file cannot be viewed.	The XML file is too large to be viewed with the Service Desk viewing tool.	The viewer does not work with large XML files. View large XML files with a text editor.

**Table 7-2 Problem Solving**

<b>Problem</b>	<b>Cause</b>	<b>Solution</b>
Errors exporting data to an XML file.	The ID field name was used in the configuration file. The sd_export program uses the automatically inserts a field called ID to give each record a unique ID in the SML files. If an additional ID field is entered by the user it will cause errors during the extraction process	Don't use the ID field, use an alternative field name, for example NNM_ID, or Event_ID.
SQL errors during export process	The export log file contains all SQL statements, if something is inaccurate in the syntax, errors may occur.	Check the syntax of the SQL statement, or copy and paste the SQL statement directly to MS Access.
Not all data in the XML file were imported.	Syntax errors in the import mapping.	Check the log file for errors related to: case, extra spaces, and spelling. Correct any errors listed. Select the Debug mode to generate a detailed log and run the process again.
Not all data in the XML file were imported.	Incorrect or incomplete import mapping	Verify that all items are mapped in Service Desk accurately then run the process again.

**Table 7-2 Problem Solving**

<b>Problem</b>	<b>Cause</b>	<b>Solution</b>
Not all items in the XML file were imported.	If the import mapping of a class that has already been mapped and imported into Service Desk is changed, to import that class to a different location in Service Desk, your data may not be imported correctly.	Always map classes to the same items in Service Desk.

Some additional tips which may help:

**Table 7-3 Tips**

Set LOADTABLE=TRUE (default) in the configuration file if you think that it will fit in the memory space available. This can speed up the export process considerably.
If you experience errors during the import process. Try importing small XML files first. This is quicker and makes it easier to locate the problem causing the error.
Use the MAXRECORDS setting in the configuration file to limit the number of records selected during export, when you are initially exporting data.
Class and attribute names in the export configuration file, XML file, and import mapping are case sensitive. The import log file will show all attributes and classes that could not be matched.
Do not use the attribute name <i>ID</i> , it is reserved for internal use only.
Do not view large XML files with the Viewer, use a text editor instead.



**Table 7-3**

**Tips**

During the export of parent-child relations, it is important to note that: Columns that appear in PARENT\_RELATION have to be put into the column list when tables are cached [LOADTABLE=TRUE], because the extractor uses this information to find all children for a particular parent in memory.

Auditing and Troubleshooting  
**Troubleshooting**

---

# **A Integration With Network Node Manager**

The Service Desk integration with NNM is a one way integration for importing service events and node information into Service Desk.

## Integration Possibilities

Network Node Manager (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:

- automatically send events from NNM to Service Desk;
- import NNM Nodes into Service Desk as Configuration Items.

### Creating Incidents in Service Desk

You can use `sd_event.exe` to automatically send event information to Service Desk. Events that can be sent from NNM to Service Desk include:

- Insert an incident with *OV\_Node\_Down*
- Insert a configuration item with *OV\_Node\_Add*
- Insert an incident with *OV\_Segment\_Critical*
- Insert an incident with *OV\_Network\_Critical*

### Importing NNM Nodes into Service Desk

Node information from Network Node Manager can be extracted and imported into Service Desk as configuration items. A Data Exchange task can be created to extract the information from Network Node Manager and import it into Service Desk. The extraction is done via an open database connectivity link (ODBC) to the NNM database. The data will first need to be exported to an Oracle database, see “Special Instructions for Network Node Manager” on page 47 for additional information on creating an Oracle account for exporting NNM node information using ODBC.

---

## Installation

This section provides an overview of the integration installation. For detailed installation instructions, refer to the Service Desk Installation Guide.

### Service Desk Application Server

After installing the Service Desk application server, you will need to install *Integrations*, with the Data Exchange and NNM options selected, on the Service Desk application server from the Service Desk CD-ROM. The following tools and files will be installed:

**Table A-1 Service Desk Server**

<b>Installed Items</b>	<b>Location</b>	<b>Status</b>
sd_event.exe	service desk path\server\bin	ready
external_event (import mapping)	Service Desk database (Can be configured from the client)	default values
sd_event.ini	Service Desk path\server\bin	default values

## Configuration

The Network Node Manager integration with Service Desk can be configured in the graphical user interface of that application, or from the `trapd.conf` file.

### Configuring Network Node Manager

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

#### Select an NNM Event

Open NNM and select an event.

1. Start the Network Node Manager Services.
2. Start the Network Management Console.
3. In NNM file menu select `Event Configuration` from the `Options` menu.
4. In the `Enterprises` list, click `OpenView`. The `Events for Enterprise` window will show events that are generated by NNM processes.

As an alternative, you can work directly from the `conf/C/trapd.conf` file. By adding the `EXEC` keyword to the `trapd.conf` file you can pass an incident or a newly detected configuration item to Service Desk.

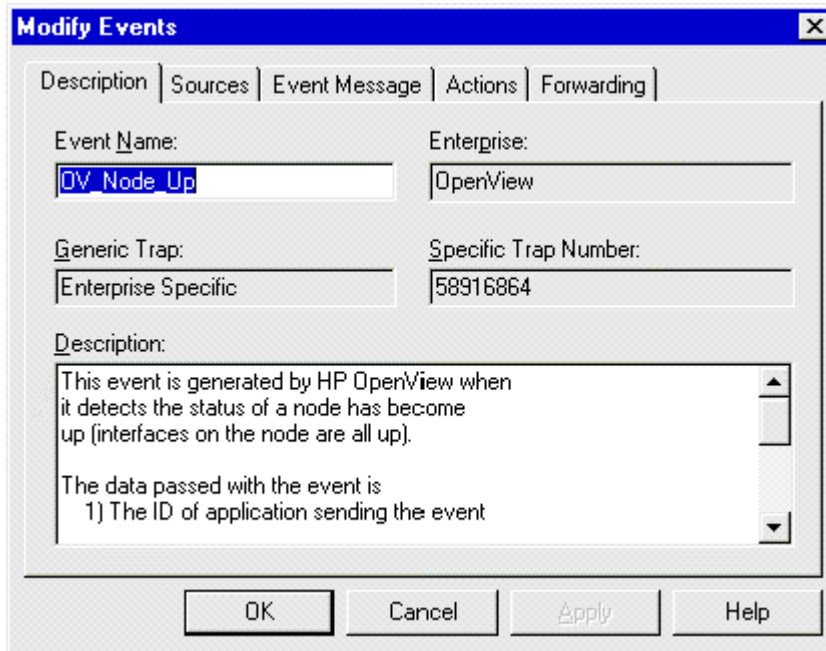
#### Modify an Event

Modify the event in NNM:

1. Double-click one of the events from the `Events for Enterprise OpenView` window, for example `OV_Node_Up`.

The `Modify Events` dialog box appears:

**Figure A-1 Modify an Event in NNM**

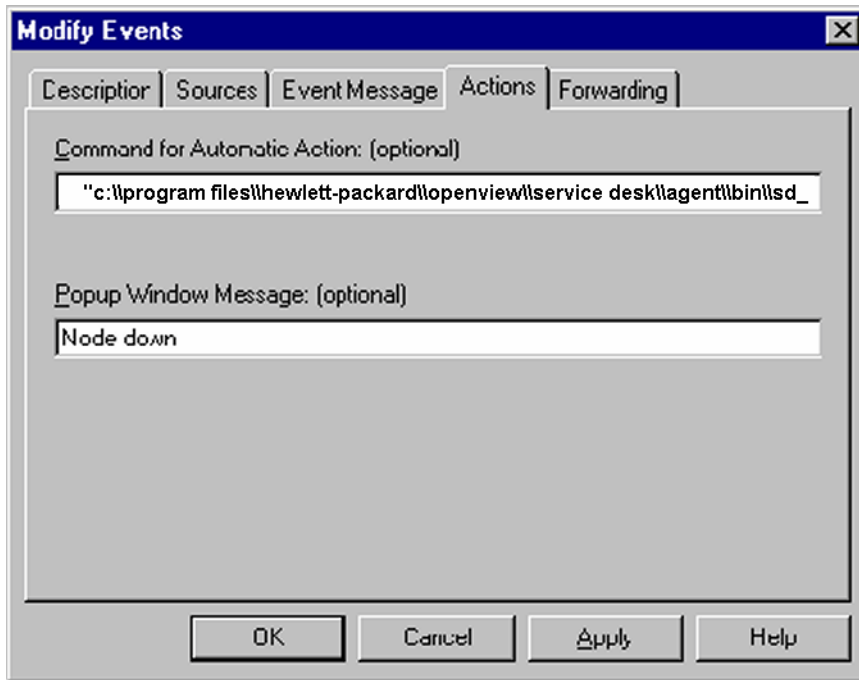


2. In the NNM application, click the Actions tab and enter the command line in the Command for Automatic Action field. The complete command line will look similar to:

```
"C:\\program files\\hewlett-packard\\openview\\service  
desk 3.0\\client\\bin\\sd_event.exe" -f  
"C:\\program files\\hewlett-packard\\openview\\service  
desk 3.0\\client\\bin\\sd_event.ini" -v  
event_id= "$2 $x $X" description= "$2 went down at $x  
$X"
```

where client can be either client or server.

**Figure A-2 Service Event Command Line in NNM**



3. Click OK.
4. From the File menu click Save and then Close, to save and activate the new settings in NNM.

**NOTE**

Actions in NNM are executed by an action daemon. If the syntax of the action command line is not correct, error messages will appear in the NNM 6 directory at: log\ovactiond.log.

### **Configuring Service Desk**

This section explains the configuration steps that must be done in the Service Desk application



## Configure the Sd\_event File

From the Service Desk 3.0\server\bin folder open the `sd_event.ini` file, rename it and configure it as follows:

1. In the `Server` line enter the name of your Service Desk application server.
2. After `Account` enter the account created for this service event integration and the password.
3. Additional items which need to be configured can be seen in the example `sd_event.ini` file shown below with default values. For additional information on using the configuration file, refer to the *HP OpenView Data Exchange Administrator's Guide*:

```
[SD_Event]
LOGFILE=c:\temp\sd_event.log
ERROR_LOGFILE=c:\temp\sd_event_error.log
ACCOUNT=nnm6_event/servicedesk
SERVER=localhost
PORT=30980
MAPPING=external_event
CLASSNAME=incident
MODUS=insert
LANGUAGE=GB
```

---

### NOTE

You can open configurable ini files for editing from the Data Exchange dialog box by entering the ini file name, then click Edit. The file will open in a text editor.

---

## Modify the Import Mapping

In Service Desk, modify the `external_event` import mapping as needed.

## NNM Variables

A number of variables are available for use in NNM, which enable you to control the formatted output sent from NNM to Service Desk. Special variables can be used in the Event Log Message, Popup Window Message and Command for Automatic Action of the Add Event,

Modify Events or Copy Event fields.

### Special Characters

Standard `Cprintf` formats will be converted to the equivalent ASCII format. All non-printable characters will be converted to the octal(`\ooo`) equivalent for display in the event browser, `trapd.log`, and when passed to the operator (manual) actions. The two exceptions are that a tab is displayed as `\t` in the event browser and `trapd.log` and as a space in pop-up messages. A new line is displayed as `\n` in the event browser and `trapd.log` and as a new line in pop-up messages. All non-printable characters are passed in their original form to automatic actions executed by `ovactiond`.

### Sequential Attribute Variables

The `$` variables, shown in the following table, are used to access the sequential attributes that were received with the event. Each event has associated attributes, however the event can have no associated attributes. They are accessed using the `$n` notation, where `n` is the positional attribute with 1 being the first attribute. The printing format is based on the ASN1 type of the attribute. These attributes are equivalent to the variable bindings (`varBinds`) in an SNMP trap. The variables are as follows:

**Table A-2 Sequential Attribute Variables**

Variable	Definition
<code>\$#</code>	Print the number of attributes in the event.
<code>\$*</code>	Print all the attributes as seq name (type): value strings, where seq is the attribute sequence number.
<code>\$n</code>	Print the nth attribute as a value string. It must be in the range of 1 to 99.
<code>\$-n</code>	Print the nth attribute as a seq name (type): value string. It must be in the range of 1 to 99.
<code>\$+n</code>	Print the nth attribute as a name: value string. It must be in the range of 1 to 99.

**Table A-2 Sequential Attribute Variables**

Variable	Definition
\$>n	Print all attributes greater than n as value strings. This is useful for printing a variable number of arguments. \$>0 is equivalent to \$* without sequence numbers, names or types.
\$>-n	Print all attributes greater than n as seq name (type): value strings.
\$>+n	Print all variables greater than n as a name: value strings.

**Special Information Variables**

You can also include information from the incoming event by using the \$arg format specification. The following \$ variables are valid regardless of the type of event (SNMPv1, SNMPv2C, CMIP, GENERIC).

**Table A-3 Special Information Variables**

Variable	Definition
\$x	print the date the event was received using the local date representation.
\$X	Print the time the event was received using the local time representation.
\$@	Print the time the event was received as a number of seconds since the Epoch (Jan 1, 1970) using the time_t representation.
\$O	Print the name (object identifier) of the received event.
\$o	Print the name (object identifier) of the received event as a string of numbers.
\$V	Print the event type, based on how the event was transported. See NNM documentation for a list of supported types.

**Table A-3 Special Information Variables**

<b>Variable</b>	<b>Definition</b>
\$r	Print the implied “source” of the event in text format. This may not be the true source of the event if the true source is proxy for another source, such as when a monitoring application running locally is reporting information about a remote node.
\$R	Print the true source of the event in text format. This value is inferred by means of the mechanism that delivered the event. If the event was forwarded, this will display the address of the remote event framework.
\$c	Print the category the event belongs in.
\$s	Print the severity of the event.
\$N	Print the name (textual alias) of the event format specification used to format the event, as defined in trapd.conf.
\$F	Print the textual name of the remote event’s machine, if the event was forwarded, otherwise print the local machine’s name.
\$\$	Print the \$ character.

**SNMP-Specific Variables**

The following variables are meaningful for events created from SNMPv1 or SNMPv2 traps/informs:

**Table A-4 SNMP Specific Variables**

<b>Variable</b>	<b>Definition</b>
SC	Print the trap community string. Set to public for non-SNMPv1 events.

**Table A-4 SNMP Specific Variables**

Variable	Definition
\$E	Print the trap enterprise as a text string if possible, otherwise as in the \$e argument below. This option tries to use the enterprise name as formatted in trapd.conf, as opposed to \$O which formats using the MIB definitions, (typically a longer string). The event object identifier for non-SNMPv1 events implies this number.
\$e	print the trap enterprise as an Object ID string of numbers. The event object identifier for non-SNMPv1 events implies this number.
\$A	Print the trap agent addresses as defined in the trap PDU. This may be different from the agent that actually sent the event. If the name server knows about this node, the node name will be printed, otherwise the address will be printed.
\$G	Print the trap's generic trap number. The event object identifier for non-SNMPv1 events implies this number.
\$S	Print the trap's specific trap number. The event object identifier for non-SNMPv1 events implies this number.
\$T	Print the trap's sysUpTime time stamp. This is the remote machine's time in hundredths of a second between the last initialization of the device and the generation of the trap. It is not the time the event was received (see \$s, \$X, and \$@). For non-SNMPv1 events this value is 0.

---

## Example of NNM on UNIX Server

If you are running NNM on a UNIX server you will need to install the items mentioned in “Service Desk Application Server” on page 133, and you will also need to install the Event Communicator from the HPOVSD depot on your NNM for UNIX application server. For instructions on installing the HPOVSD depot files, refer to the Service Desk Installation Guide.

**Table A-5 NNM UNIX Server**

<b>File</b>	<b>Location</b>	<b>Status</b>
sd_event.ini	NNM server opt/OV/SD/bin	default values
sd_event	NNM server /opt/OV/SD/bin	ready

## Example of NNM on UNIX

It is possible to send service events from NNM on a UNIX system to Service Desk. The following example is for the OV\_Node\_Down event. From the Options menu in your NNM application:

1. Select Event Configuration, and then OpenView in the dialog box that opens.
2. Scroll down the Event Name list and double-click OV\_Node\_Down.
3. The following command and parameters can be used:  

```
cd /opt/OV/SD/bin; /opt/perl5/bin/perl  
/opt/OV/SD/bin/sd_event -f /opt/OV/SD/bin/sd_event.ini -v  
event_id="$2 $x $X" description="$2 went down at $x $X"  
ci=$2
```

## Managing Event Storms on UNIX

You can install the queuing tools from the HPOVSD depot file to handle event storms. The queuing tools are not mandatory, the integration will work fine without installing them. See “Preventing Event Storms” on page 107 for additional information on the queuing tools available with

Service Desk. Refer to the *HP OpenView Service Desk: Installation Guide* for installation instructions.

Integration With Network Node Manager  
**Example of NNM on UNIX Server**



---

## **B** Integration With ITO

The bi-directional event interface with ITO uses `sd_event` and Rule Manager features present in Service Desk. The service event program, `sd_event`, is used to forward ITO messages to Service Desk, and the Rule Manager is used to send annotations and acknowledgments back.

## Integration Possibilities

The Service Desk integration with ITO consists of:

- automatic forwarding of messages to Service Desk from ITO;
- automatic forwarding of acknowledgment messages from Service Desk to ITO;
- automatic forwarding of message annotations from Service Desk to ITO;
- manual forwarding of messages to Service Desk.

### Example B-1

#### ITO Integration Example

The integration can be set up to work like the following example:

1. A fault is detected in the managed network and a message is created in ITO.
2. `ito_server1` sends the message as a service event with account `ito_server1` to Service Desk.
3. Service Desk receives the event and creates an incident. The ITO message ID is imported to the `Source ID` field in the incident.
4. The Rule Manager detects the insertion of a new incident in the database and checks if it is an ITO event.
5. The Rule Manager initiates the action to send a confirmation annotation with: `opcaddanno opc_adm`. The annotation includes the `source ID` and it is sent to the agent on `ito_server1` with the message: This message is registered in Service Desk with number ID.
6. The Rule Manager agent on `ito_server1` receives the annotation and adds it to the message with that `source ID` in ITO.
7. A Service Desk user handles the incident and puts the incident in a `Closed` status.
8. The Rule Manager detects that the incident has changed in the database to the status `Closed`.
9. The Rule Manager initiates the action to send an acknowledgment message with: `opcackmsg source ID` is sent to the agent on

`ito_server1.`

10. The agent on `ito_server1` receives the acknowledgment and closes the message.

## Automatic Forwarding of Messages

You can use the trouble ticket interface in ITO to send event information to Service Desk. This makes it possible to update incident attributes in Service Desk when the event status changes in ITO. The trouble ticket interface calls `sd_eventins.sh`, the call forwarding executable, to perform this action. The message source templates in ITO need to be configured to specify which messages will be sent to Service Desk.

## Automatic Forwarding of Acknowledgment Messages

When an incident is closed in Service Desk an acknowledgment message can be sent to ITO. When the acknowledgment is received in ITO the related message is removed from the desktop and filed. Typically acknowledgments are given in ITO when:

- work has been finished on the ITO message and related problems are resolved;
- there is another message in the message browser describing the same event;
- you no longer need the message, because it has a low severity and requires no action.

Service Desk uses database rules to send acknowledgments. Example rules containing the default parameters for this action are available in the demo database.

## Automatic Forwarding of Message Annotations

Annotations are used in ITO for adding reference information to a message. Message annotations can be provide information on the:

- actions performed to solve the problem;
- name of the user who started the action;
- status of the action, performed;
- start and finish time of the action.

Service Desk uses database rules to send annotations to ITO. The demo database includes example rules for sending an annotation to ITO when an incident is created from an ITO message. Database rules can also be used to send an annotation to ITO when the status of that incident changes.

## **Manually Forwarding Messages**

This feature makes it possible to forward important messages that have not yet reached Service Desk or that failed to reach Service Desk because of problems in the automatic interface.

`sd_eventins.sh` is automatically inserted on your ITO server during the installation process. This script makes it possible to manually forward messages to Service Desk with the Application Bank in ITO. To manually forward messages with the Application Bank in ITO:

1. Start the Message Browser in ITO.
2. Open the ITO Application Bank.
3. Click the Service Desk icon (created during installation).
4. Select the items you want to send and click the Insert Incident icon. The selected item will be sent to Service Desk.

## Installation

This chapter explains how to install the tools and files necessary for the ITO integration with Service Desk to work.

### NOTE

The correct version of Perl must be installed prior to installing the ITO integration tools, and must be mentioned first in the path variables. The version of Perl that has been tested and is known to work with Service Desk ITO integrations can be found in the *Supported Platforms List*. The most up-to-date version of this list can be found in the latest HP OpenView Service Desk Service Pack under the file name `\\Doc\Supported_Platforms_List.pdf`.

## Service Desk

For the ITO integration you will need to install some items on your Service Desk application server and other items on your ITO server. The steps that follow are to be performed on the Service Desk server, see “ITO” on page 150 for information on completing the installation on your ITO server.

### Integrations

On the Service Desk server you will need to install Integrations from the Service Desk 3.0 CD-ROM. Refer to the *HP OpenView Service Desk 3.0: Installation Guide* for additional installation information.

The following tools and files will be installed:

**Table B-1**

### Service Desk Server

Installed Items	Location	Remarks
external_event	Service Desk database	import mapping
ito_acknowledgment	Service Desk database	database rule
ito_annotate	Service Desk database	database rule

### **Demo Database**

The demo database contains configured database rules for sending acknowledgments and annotations. If you do not want to install the demo database and make use of the configured rules you can create them manually. For additional information on manually configuring the database rules see “Database Rules” on page 161.

For additional installation information on installing the demo database, refer to the *HP OpenView Service Desk 3.0 Installation Guide*.

### **ITO**

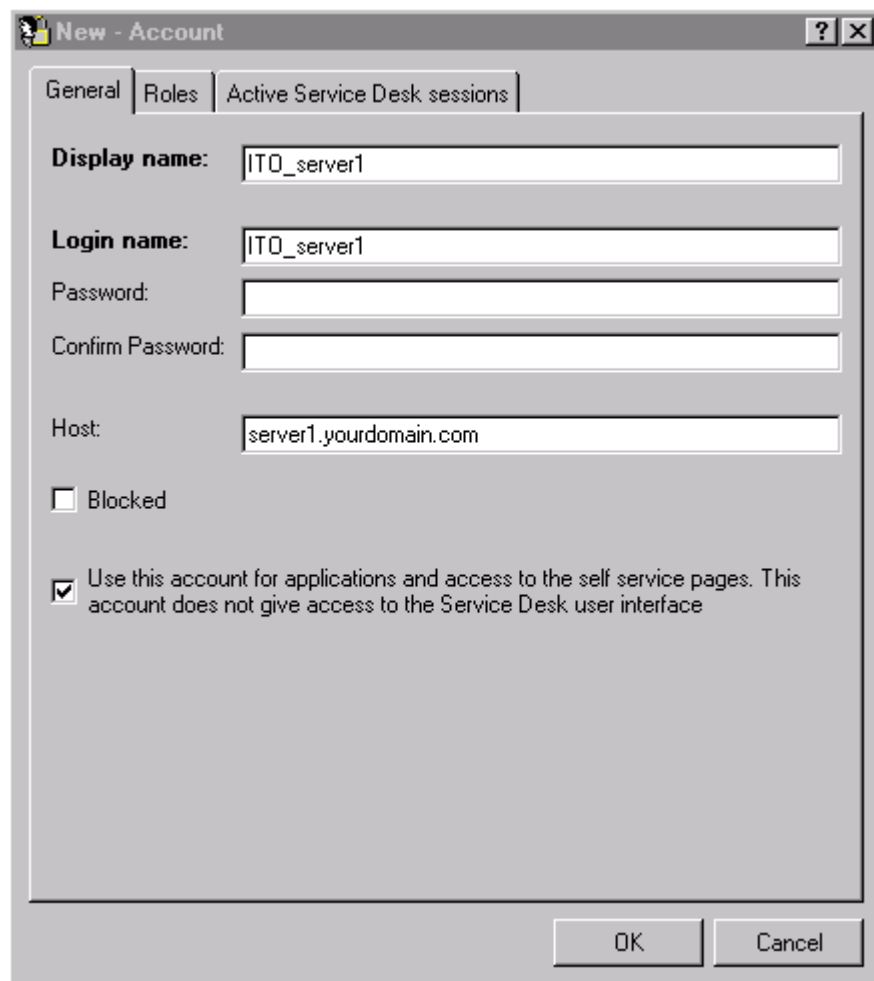
Some files and tools need to be installed on the ITO servers for the integration to work.

### **ITO Server Account**

Before installing the integration on the ITO server, create an account for each ITO server you will integrate with Service Desk, start the account name with `ito_` followed by your server name, for example `ito_yourITOserver`.

1. Make the account a non-user interface account, by selecting the Use this account for applications and access to the self service pages. This account does not give access to the Service Desk user interface **check box**.
2. Assign each account a helpdesk user role, as a minimum. See “Integration Accounts” in Chapter 4 of the *Data Exchange Administrator's Guide* for more information about accounts. The following Account dialog box is provided as an example:

**Figure B-1 ITO Account dialog box**



### The HPOVSD Depot

Install additional integration tools from the `hpoovsd` depot on your UNIX® or Solaris ITO servers. The `hpoovsd` depot is located in the `/unix` folder on the Service Desk 3.0 CD-ROM. See the *HP OpenView Service Desk 3.0: Installation Guide* for more detailed installation instructions. The installation is briefly run as follows:

1. The `hpoovsd` depot contains the ITO tools in a tar file called

## Installation

ito5Integration. The hpovsd depot also contains SDevent, EventQueuing and the Agent. To install the ITO tar file, copy hpovsd.depot to your tmp directory and then run:

- # swinstall -s /tmp/hpovsd.depot
- # mkdir /opt/OV/SD/ito5
- # cp ITO5\_SD30.tar /opt/OV/SD/ito5
- # cd /opt/OV/SD/ito5
- # tar xvof ito5\_SD30.tar
- # ./install.sh

2. During the installation of the ITO tar file, you will be asked a number of questions similar to the following:

<b>Question</b>	<b>Remark</b>
What is the Oracle version 7 or 8?	Enter the Oracle version you are using: 7 or 8.
What is the Service Desk server?	Enter the server you installed Service Desk on.
What is your ITO account?	Enter the account you have created in Service Desk for the ITO integration.
What is the database instance name of IT/operations?	Enter the alias name of the database ITO uses. The default is openview.
Do you want to load IT/Operations default configuration?	Yes installs all items from the tar file. No installs the files but does not install the ITO user interface items mentioned directly after.

3. During installation, a log file called install.log, will be created in the log folder. When the installation is successful, you will see "Installation of ITO integration has been completed without errors".



The following files will be installed automatically on your ITO server:

**Table B-2**

**ITO server**

<b>File</b>	<b>Location</b>
<code>sd_event.ini</code>	<code>/opt/OV/SD/bin</code>
<code>sd_event.sh</code>	<code>/opt/OV/bin/OpC/extern_intf</code>
<code>sd_eventins.sh</code>	<code>/opt/OV/bin/OpC/extern_intf</code>
<code>sd_eventins.pl</code>	<code>/opt/OV/bin/OpC/extern_intf</code>
<code>get_ITO_attributes</code>	<code>/opt/OV/bin/OpC/extern_intf</code>
<code>opcaddanno</code>	<code>/opt/OV/bin/OpC/extern_intf</code>

The installation will also automatically:

- adapt the `sd_event.ini` file located in `/opt/OV/SD/bin`;
- add the trouble ticket setting `sd_event.sh` to the ITO user interface;
- add the application group `Service Desk` with application `Insert Incident` in the ITO user interface;
- add message group `Service Desk` to the ITO user interface;
- add message template group `ServiceDesk` with two message templates:

<code>SD_ITO</code>	Detects errors and warnings in the <code>ITO_SD</code> event integration.
<code>SD_ITO_ACK</code>	Detects if the Rule Manager agent cannot acknowledge messages.

Messages created from the templates are stored in the `Service Desk` message group.

**NOTE**

The installation procedure also creates a log file (`install.log`) in the `log` directory.

### Multiple Servers

To support multiple servers you will need to:

- create a separate account in Service Desk for each ITO server and specify the host name for each account for sending acknowledgment and annotation messages to the proper ITO server. For example:

<b>Account</b>	<b>Host</b>
ito_account1	server1.yourdomain.com
ito_account2	server2.yourdomain.com

- install the ITO integration on each server;
- install the Service Desk Agent and queuing tools on each server.

## Configuration

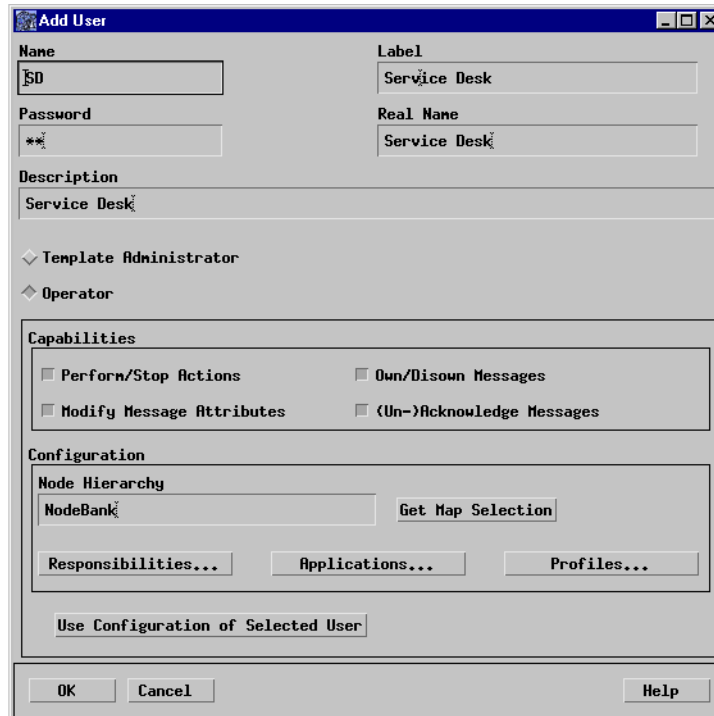
This section explains the configuration steps that need to be performed in Service Desk and in ITO.

### Make Service Desk an ITO User

Make Service Desk an ITO user called SD. The commands `sd_event.sh` and `sd_eventins.sh` create new incidents. These commands use the SD user in ITO to gather information from ITO. The SD operator in ITO owns the messages as soon as they are forwarded to Service Desk, making it possible to distinguish ITO messages from other messages in Service Desk.

1. From the Window menu in ITO, select User bank to manually create a user.

**Figure B-2 Add User dialog box**



2. In the Name field enter SD and give it the password sd.
3. Enter Service Desk in the Label, Real Name and Description fields.
4. Select the Operator option for the SD user.
5. Click Responsibilities and select HP-UX to select all responsibilities.
6. Click OK when finished.

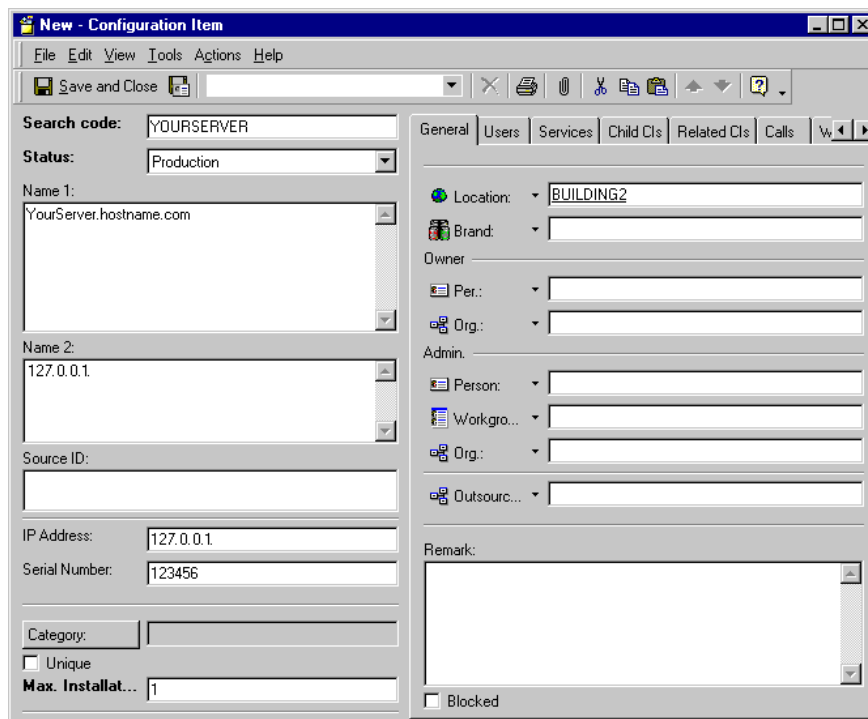
## Create Configuration Items

In Service Desk, create configuration items for all nodes managed by ITO. This step needs to be performed so that events coming from the ITO server can find the configuration item they are related to in Service Desk. The ITO server is also a managed node. In the Configuration Item dialog box the Search code, Name 1 and Name 2 fields are mandatory.

You can create configuration items for your nodes as follows:

1. Open a new Configuration Item dialog box. This can be done by selecting Configuration Item from the shortcut bar or selecting New then Configuration Item from the File menu.
2. In the Search code field enter the name of the managed node.
3. In the Name 1 field enter the complete host name for the managed node.
4. In the Name 2 field enter the IP address for the managed node. The Configuration Item dialog box will look similar to the following:

**Figure B-3 Configuration Item dialog box**



**TIP**

It is possible to use the Data Exchange integration with Network Node Manager to import managed nodes as configuration items in Service

Desk, instead of manually entering them in Service Desk.

## Import Mapping

In Service Desk, modify the import mapping as needed. An import mapping called `external_event` is provided with default mapping; you can modify it as needed. The event interface uses the ITO API to get message details. This provides access to 48 message attributes in ITO, for example, message ID, message text, instructions, and annotations. You can configure the `sd_eventins.pl` file to put ITO message attributes in the Remark and Information fields of an incident in Service Desk. For example:

```
information="\Application: $ito_params{APPLICATION}; Object: $ito_params{OBJECT}\".
```

An incident created by ITO in Service Desk will have the following attributes by default:

Event ID	ITO message ID
Description	ITO message text
Information	“Original message: <message text>” “Detected by application: <message application>” “Object in question: <message object>” “Annotations: <annotations>”
Solution	ITO instructions
CI	ITO message node name
Impact	ITO severity
Status	ITO message status

For additional information about how to conduct import mapping, see Chapter 3 on “Mapping Service Events” in the *Data Exchange Administrator's Guide*.

## Attribute Mapping

ITO message attributes are mapped to Service Desk incident attributes. The following table contains descriptive information to assist you in

modifying the mapping supplied with Service Desk:

**Table B-3 Service Desk Attributes**

<b>Attribute</b>	<b>Description</b>
Configuration Item	The configuration item in Service Desk that corresponds to the ITO node or object that the error occurred on.
Description (Mandatory field)	The ITO <message text>
Information	The text: Original message: <message text> Detected by application: <name of application> Object in question: <object name> Annotations: <annotations>
Solution	<instruction text>
Category (default)	To specify the category, choose the ITO incident template and edit the category field.
Status (default)	The status the incident will receive when created. To specify the status, choose the ITO incident template and edit the Status field.
Impact (default)	The priority code is based on the ITO <message severity> if configured that way. To map the message severity, choose the ITO import mapping template. Map the message severity using the Value Mapping options in Data Exchange.
Assigned to person (default)	The person the incident will be assigned to. To specify a specialist, choose the ITO incident template and edit the Specialist field.
Assigned to group (default)	The group the incident will be dispatched to. Must be a group the assigned specialist belongs to.
Pool (default)	The pool the incident will be placed in. The ITO integration uses the system user's pool as the default pool for storing incidents.

For additional information about ITO message attributes, refer to the *IT/Operations Developer's Toolkit: Application Integration Guide*.

## The Event Queue

Configure the queuing tools if you expect to experience event storms. See Chapter 5 “Importing Service Events” in the *Data Exchange Administrator's Guide* for more information. The event queuing tools can be installed from the HPOVSD depot. An example of how to configure the event queue for ITO follows:

### Example B-2

#### Configuring the Event Queue for the ITO Integration

```
# mkdir/opt/OV/bin/OpC/extern_intf/sd_event_queue
```

To use the `addentry.sh` call, modify `sd_event.sh` so that it looks like the following example:

```
cd /opt/OV/SD/queuing
/opt/OV/SD/queuing/addentry.sh
/opt/OV/bin/OpC/extern_intf/sd_event_queue
"cd/opt/OV/SD/bin;
/opt/OV/bin/OpC/extern_intf/get_ito_attributes SD sd
$ITO_Message|/opt/OV/bin/OpC/extern_intf/sd_eventins.pl"
```

#### Configure the Ini File

Configure the `sd_event.ini` file. It can be found in the Service Desk `/opt/OV/SD/bin` folder. Many of the items will be configured automatically for you during the installation process. You will want to enter the name of the import mapping you configured, in the MAPPING row. An example configuration file follows:

```
[SD_Event]
LOGFILE=sd_event.log
ERROR_LOGFILE=sd_event_error.log
ACCOUNT=ITO_server1/ITO_server1
SERVER=local host
PORT=30980
MAPPING=external_event
CLASSNAME=incident
MODUS=insert
LANGUAGE=GB
```



## Configure the Trouble Ticket Interface

Configure the ITO Trouble Ticket Interface:

- `sd_event.sh` will be set as the default call forwarding executable during the installation process. It uses `sd_event.pl` to call `sd_event` from `/opt/OV/SD/bin`.
- The Message Source Templates, listing what items are to be sent to Service Desk are *not set* by default. Configure the templates to send items from ITO to Service Desk. The ITO incident template can be found by clicking `Data, Templates` then `Incident` in the Administrator Console.

## Database Rules

Database rules are used to send information from Service Desk to ITO. A Service Desk agent must be running on each ITO server to execute the commands specified in the rules.

The following rules for sending acknowledgments and annotations to ITO for UNIX servers are available in the demo database. The following examples are provided to help you create the rules manually if you have decided not to install the demo database. You will need to fill the bulleted information in the database rules wizard:

- Send acknowledgments to ITO for UNIX:
  - for each modified incident;
  - if account name of the registration starts with ITO;
  - and status is changed to closed;
  - then execute `/opt/OV/bin/OpC/opcackmsg[Source Id]`.
- Send annotations to ITO for UNIX (uses `opcaddanno`):
  - for each inserted incident;
  - if account name of the registration starts with ITO;
  - then execute `/opt/OV/bin/OpC/opcaddanno [Source ID]. A Service Desk incident has been created with number ID.`;
  - on host `[Registration; Created by;Host]`.

---

**NOTE**

`opcackmsg` is not installed by Service Desk; it is part of the ITO application and is located in `/opt/ov/bin/OpC`.

---

**Creating New Database Rules**

You can create additional database rules as needed. To create a new rule:

1. From the `Tools` menu in Service Desk select `System`, then click `Business Logic`, then `Database Rules`.
2. Double-click the `Incident` to open the `Database Rules Wizard`.
3. Right-click and select `New database rule` from the menu that appears.

---

**TIP**

Create new database rules quickly by using the copy and paste functions in the Rule Manager. Select a rule that is similar to the rule you want to create and click `CTRL+C` then `CTRL+V` to make a copy. Double-click the copied rule to open it and then use the Rule Wizard to change the parameters and the name.

---

---

# **C** **Integration With ManageX**

This appendix provides information useful in setting up an integration with ManageX for the purpose of sending service events to Service Desk and sending service events from Service Desk back to ManageX.

## **Integration Possibilities**

The ManageX Management Console allows system administrators to explore domains and machines in the network. You can observe, administer, and manage network machines, all from a central console. A performance monitor provides detailed real-time examination of performance data. Events detected by ManageX can be forwarded to Service Desk and registered as service calls so that help desk personnel and specialists can quickly react, solving the problem. The following features are available with this integration:

- Service Events detected in ManageX can be sent to Service Desk. The incoming service events are registered as incidents in Service Desk.
- Service Events can be sent from Service Desk to ManageX when an incident is created, when the status changes, or when it is closed.

## Installation

This section provides an overview of the items that must be installed for the integration to work. For installation instructions refer to the *HP OpenView Service Desk: Installation Guide*.

### Service Desk Application Server

For the ManageX integration you will need to install some items on your Service Desk application server and other items on your ManageX server.

After installing the Service Desk application server you will need to install Integrations with the Data Exchange and ManageX options selected, from the Service Desk 3.0 CD-ROM.

Installing Integrations will copy the following:

**Table C-1 Service Desk Server**

Installed Item	Location	Status
sd_event.exe	Service Desk 3.0\server\bin	ready
external_event (import mapping)	Service Desk database (Can be configured from the Data Exchange GUI)	default values
sd_event_managex.ini	HP OpenView\ManageX \Policies	default values
ServiceDeskSamplePas sThru.mxc	ManageX\Policies	default values
sendmessage.exe	Service Desk 3.0\server\bin	ready

### Demo Database

The demo database contains preconfigured database rules for sending service events.

The following table provides an overview of the rules available in the demo database:

**Table C-2 Service Desk Server**

Database Rules	Location	Remarks
ManageX_acknowledge	database rules in application	for acknowledging ManageX messages.
ManageX_annotate	database rule in application	for adding annotations to ManageX messages.

For additional installation information, refer to the *HP OpenView Service Desk 3.0 Installation Guide*.

### ManageX Application Server

If ManageX is running on the same server as Service Desk, the files and tools needed for the integration will be installed when you install Integrations from the Service Desk CD-ROM and the ManageX option is selected.

The ManageX lights out policy called:

`ServiceDeskSamplePassThru.mxc` will be installed in `C:\Program Files\HP OpenView ManageX\Policies`. You will need to perform some additional steps to complete the installation as explained in the next section.

If ManageX is running on a server different from the ManageX server see “ManageX on a Different Server” on page 167.

### ServiceDeskSamplePassThru

`ServiceDeskSamplePassThru.mxc` is installed out-of-the box to `Service Desk\Bin` when you install ManageX from the Service Desk CD-ROM. Some user actions still need to be performed manually as follows:

1. From the ManageX console, select the ManageX Server from the Device Selector dialog box and click Apply. If you don't see the Device Selector dialog box when you start the ManageX console, right-click

- the OpenView ManageX node and choose Device selector.
2. Double-click Policies from the ManageX console tree then double-click Available Policies.
  3. Right-click in the results pane to select the policy and select All Tasks then Install from the shortcut menu.

---

**TIP**

If the Lights-Out Policy ServiceDeskSamplePassThru.mxc is not listed, right-click on the Policies folder then click All Tasks. Click Set Directory and then navigate to the directory with the ServiceDeskSamplePassThru.mxc policy and click OK.

---

### **ManageX on a Different Server**

We recommend that the ManageX console runs on the same server as the Service Desk application server. If you run ManageX on a different server you will need to perform the following additional steps:

- The sd\_event\_managex.ini file needs to be copied to the ManageX server and configured to refer to the Service Desk server name.
- Rules in the Rule Manager need to be changed, reflecting the ManageX server as the agent name.
- sd\_event.exe (event communicator), and ServiceDeskSamplePassThru.mxc will need to be copied from the Service Desk\Bin path to ManageX.

## Configuration

This section explains the configuration tasks you will need to perform in Service Desk followed by the configuration tasks you will need to perform in the ManageX application.

### Configuring Service Desk

This section explains the configuration steps that must be done in Service Desk.

#### Configure the Ini File

Copy the `sd_event.ini` file and rename it to `sd_event_managex.ini` in Service Desk. The default location of `sd_event_managex.ini` is: Service Desk 3.0\server\bin or Service Desk 3.0\client\bin

1. In the `Server` line, enter the name of your Service Desk application server.
2. After `Account` enter the account created for this service event integration and the password.
3. Additional items which need to be configured can be seen in the sample `sd_event.ini` file shown below with default values:

```
[SD_Event]
LOGFILE=c:\temp\sd_event.log
ERROR_LOGFILE=c:\temp\sd_event_error.log
ACCOUNT=managex4_event\servicedesk
SERVER=localhost
PORT=30980
MAPPING=external_event
CLASSNAME=incident
MODUS=insert
LANGUAGE=GB
```

#### Database Rules

Configure the Rule Manager in Service Desk to send events back to ManageX. A database rule will be installed when you install the ManageX integration from the Service Desk CD-ROM. You will want to adjust the default values set for the rule and turn it on. You can create a



database rule to send an acknowledgment to ManageX when an incident is created, when the status changes and when the incident is closed, for example.

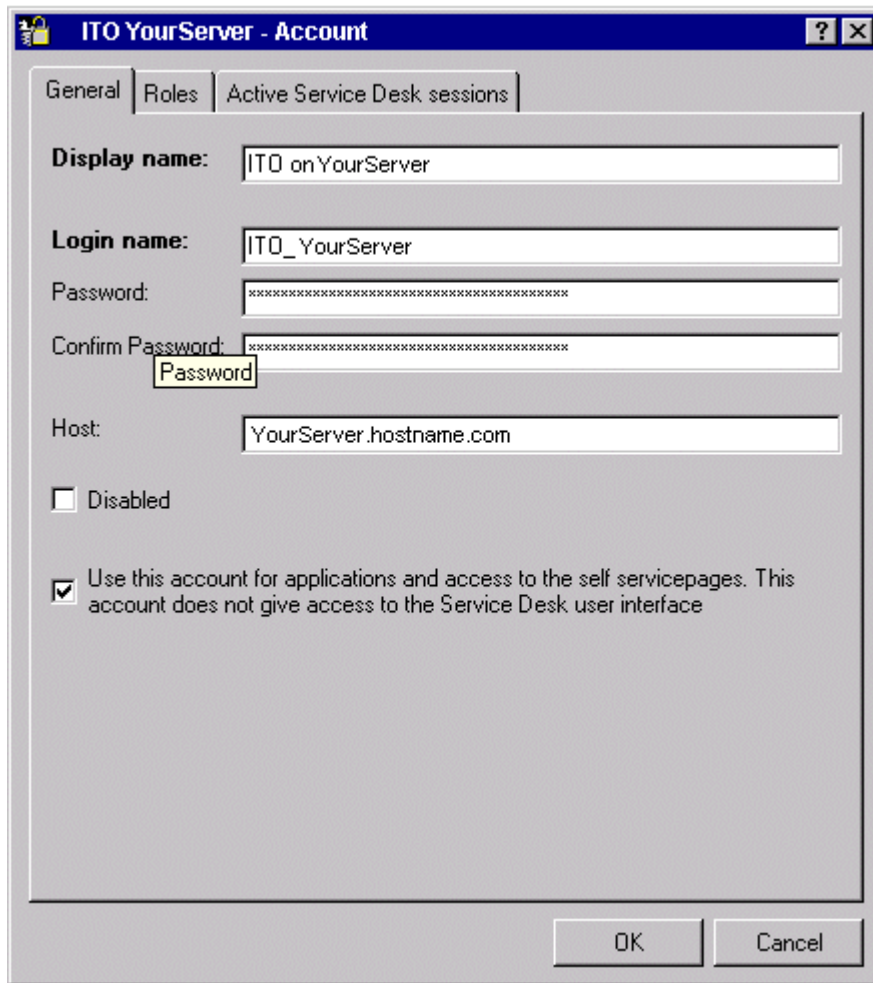
1. From the Service Desk Tools menu, click System, then Business Logic. Double-click Database Rules and select the ManageX rule.
2. Click Modify and use the Rules Wizard to make any necessary changes.
3. The Conditions should look something like:  
*specify Registration\_login=manageX4\_event  
status=closed*
4. Configure the Command Exec action to send service events to ManageX. The command line should resemble:  
*sendmessage.exe, -d="incident<ID> has been closed"*

### **ManageX Server Account**

Create an account for each ManageX server you will integrate with Service Desk:

1. Start the account name with ManageX\_ followed by your server name, for example ManageX\_yourManageXserver.
2. Make the account a non-user interface account, by selecting the Use this account for applications and access to the self service pages. This account does not give access to the Service Desk user interface check box.
3. Assign each account a helpdesk user role, as a minimum. The following Account dialog box is provided as an example:

**Figure C-1**      **ManageX Account dialog box**



## **Configuring ManageX**

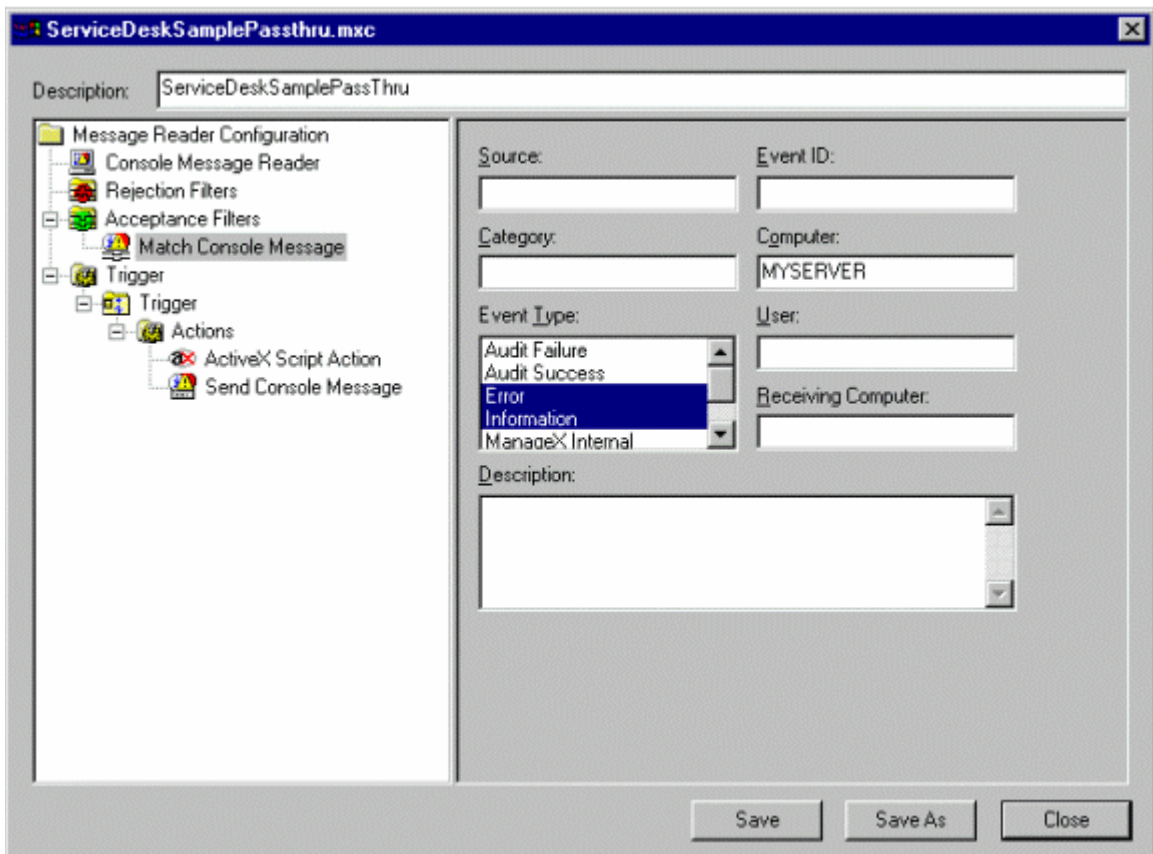
This section explains the configuration tasks you must perform in the ManageX application

### **Configure the Acceptance Filter**

Set the acceptance filter in ManageX. This will define what events are forwarded to Service Desk:

1. From the ManageX console double-click Policies, then Available, and then Lights-Out.
2. Double-click ServiceDeskSamplePassThru.mxc, and the ActiveX script action will show the mxc file in Visual Basic format. You can edit the script to send the correct event information
3. In the Message Reader Configuration folder click Acceptance Filters and then double-click Match Console Message:

**Figure C-2 Set ManageX Acceptance Filters**



4. In the results screen in the Computer field, enter the computer you want to accept events from. In the Event Type field select the type of events you want sent to Service Desk from the list.

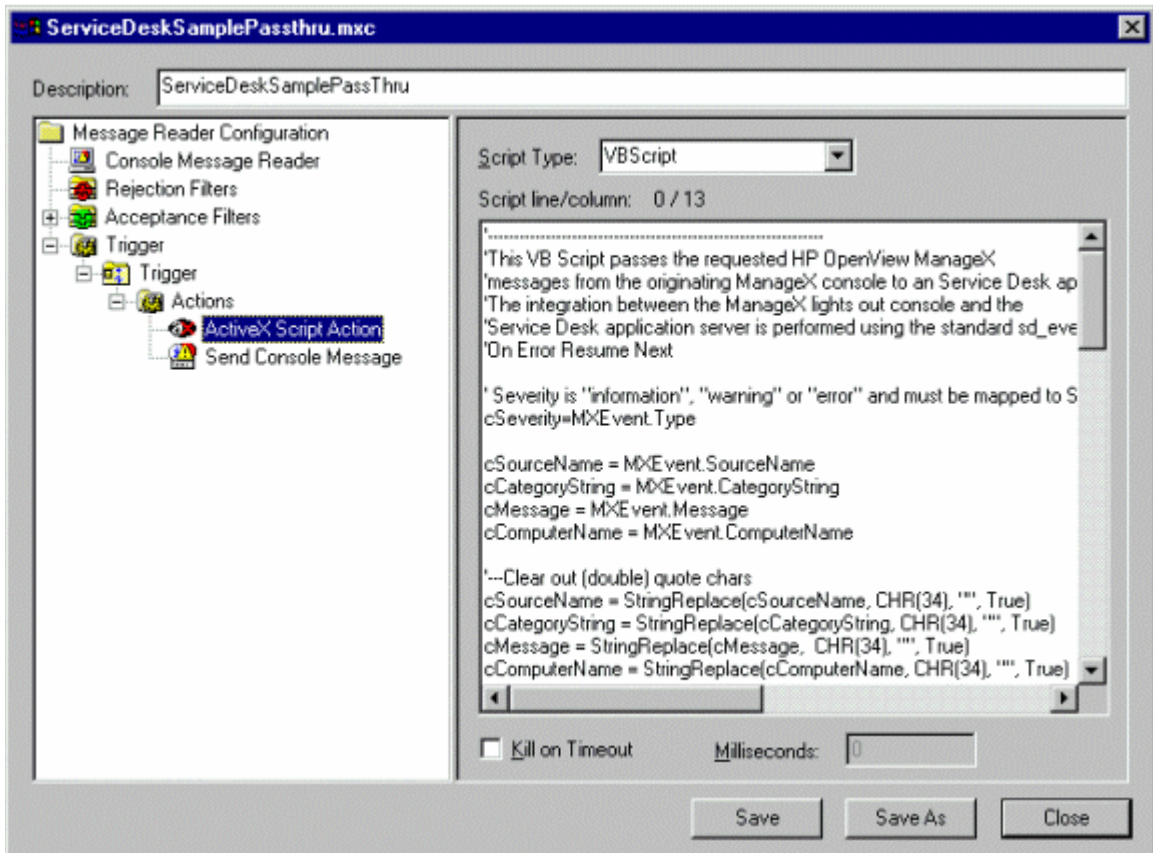
5. Click **Save** when you are finished.
6. After changing a policy you will need to install it.

### **ManageX Lights Out Policy**

Configure the ManageX Lights-Out policy. The Lights-Out policy gathers events coming from ManageX agents and sends them to Service Desk, even when the ManageX application is not running.

1. From the ManageX console double-click **Policies**, then **Available**, and then **Lights-Out**.
2. Double-click on **ServiceDeskSamplePassThru.mxc**, and **ActiveX** script action with will show the **mxo** file in Visual Basic format. You can edit the script to send the correct event information:

**Figure C-3 ManageX ActiveX Script Action**



- The lights out policy ServiceDeskSamplePassthru.mxc contains information you will need for the import mapping. The default information is as follows:  

```

cDescription = "description=" & chr(34) & cMessage & chr(34)
cCI = "ci=" & chr(34) & cComputerName & chr(34)
cInformation = "information=" & chr(34) & "Message:" & cMessage &
"\nEventid: " & "${EventID}" & chr(34)
cImpact = "impact=" & chr(34) & cSeverity & chr(34)
cClassification="classification=" & chr(34) & cCategoryString & chr(34)
cEventID= "event_id=" & chr(34) & cComputerName &
"&MxEvent.TimeGenerated & chr(34)

```

- The following incident fields in Service Desk are the most likely ones that you will map an event to:
  - event\_id (this is a required field, a unique key field)
  - description
  - information
  - solution
  - category
  - impact
  - status
  - priority
  - CI
- The following information can be used when editing the Visual Basic text to change the configuration of event messages going from ManageX to Service Desk. After changing a policy you will need to install it as explained in XXX:

**Table C-3    ManageX Event String Substitutions**

<b>String</b>	<b>Description</b>
\$(Category)	Category of the triggering Console Message.
\$(CategoryNumber)	Category of the triggering Event Log Message, only the event log.
\$(CategoryString)	Category string of the triggering Console Message.
\$(Computer)	Name of the machine that sent the triggering Message or NT Event Log event.
\$(Description)	Contents of the Description field from the triggering Console Message.
\$(EventID)	Contents of the EventID field from the triggering Console Message.
\$(Identifier)	A unique record number from the triggering Event Log message.
\$(OriginalTime)	The time the triggering Event Log item was written or the triggering Console Message was sent.

**Table C-3 ManageX Event String Substitutions**

<b>String</b>	<b>Description</b>
\$(RawDescription)	The unexpanded description of the triggering message.
\$(ReceiveComputer)	The computer receiving the triggering Console Message.
\$(ReceiveSource)	Application, Security, or System if the triggering event came from the Windows NT Application. Security OpenView Console, or System Log OpenView Console, if the trigger was an OpenView Console Message.
\$(ReceiveTime)	The time the Event Log message was read from the file or the time when the Console Message was received.
\$(Source)	The source of the triggering message.
\$(Type)	The type of triggering message. One of the following numeric values will be returned: 1=Information, 2=Warning, 3=Error, 8=Success, 16=Audit Failure.
\$(User)	The name of the user who sent the triggering message.

**Table C-4 ManageX Event COM Objects**

<b>Field/Syntax</b>	<b>Comment</b>	<b>Type</b>
CATEGORY MxEvent.Category	user-defined message category.	Unsigned Integer (UINT)
CATEGORYSTRING MxEvent.CategoryString	User-defined message category.	String
COMPUTERNAME MxEvent.ComputerName	Machine originating event.	String

**Table C-4 ManageX Event COM Objects**

<b>Field/Syntax</b>	<b>Comment</b>	<b>Type</b>
SERVICENAME MxEvent.DeviceName	User-defined identifier for device type, such as printer or hub for hardware.	String
EVENTIDENTIFIER MxEvent.EventIdentifier	User-defined integer that uniquely identifies the event.	Unsigned integer (UINT)
MESSAGE MxEvent.Message	Free-form information sent to console operator.	String
SOURCENAME MxEvent.SourceName	User-defined string identifying application	String
TIMEGENERATED MxEvent.TimeGenerated	Date field that defaults to time event was sent; user may override with a specific value.	Date
TYPE MxEvent.Type	User-defined string identifying severity level of the event; typically includes Information, Warning, and Error. Other strings may be defined as appropriate.	String
USER MxEvent.User	User-defined string naming the user responsible for generating the event	String
HELPURL MxEvent.HelpURL	String pointing to a location with additional information to assist you in dealing with the event	String

### **The Event Queue**

Configure the queuing tools if you expect to experience event storms. An example for ManageX follows:

#### **Example C-1**

#### **Configuring the Event Queue for the ManageX Integration**

The job queue executables for ManageX will be installed automatically by the installation program:



To use the `addentry.sh` call, modify `sd_event.sh` so that it looks like the following example:

```
cd /opt/OV/SD/event_queuing  
/opt/OV/SD/event_queuing/addentry.sh  
/opt/OV/bin/OpC/extern_intf/sd_event_queue  
"cd/opt/OV/SD/bin;  
/opt/OV/bin/OpC/extern_intf/get_ManageX_attributes SD sd  
$ManageX_Message|/opt/OV/bin/OpC/extern_intf/sd_eventins.pl"
```

Integration With ManageX  
**Configuration**

---

# **D** **Examples**

This appendix is designed to provide you with useful examples. The examples include how to use an example configuration file, how to import data from an MS Excel spreadsheet, and from an ASCII text file.

## Using an Example Configuration File

You will need to adapt the configuration files to fit your environment. The configuration files contains the knowledge about what data to extract and about the source you will extract data from.

If you have added or changed the classes or fields in your external application, you will need to modify the export mapping in the example configurable extractor and the import mapping in Service Desk. In some cases it may also be necessary to use the customizing features in Service Desk to create new fields to import your external fields to. For example, a field present in your application may not be included in the configuration file. In that case you will need to adjust the NNM.ini file and the NNM import mapping.

The following example will be used to explain the different parts of the configuration file:

- Step 1.** Copy an existing configuration file and save it with a new name. For this example a portion of the NNM6 example configuration file is used:

```
[DSN]
NAME=nnm6
USR=testingdb
PWD=testingdb

[SYSTEM]
LOG=TRUE
XML=TRUE
DUMP=TRUE
TXT=FALSE
LOG_FILE=C:\temp\NNM6.log
OUTPUT_FILE=c:\temp\NNM6.txt
XML_OUTPUT_FILE=c:\temp\NNM6.xml
APPLICATION_NAME=NNM6
ENCODING=ISO-8859-1
[CLASSES]
NAME=NETWORK, SEGMENT

[NETWORK]
SOURCE=[NNM_NETWORKS], [NNM_OBJECTS]
ATT=[NAME], [INTERFACE_COUNT], [NNM_ID]
COLUMNS=[NNM_NETWORKS].[IP_NETWORK_NAME] AS
[NAME], [NNM_NETWORKS].[TOPM_INTERFACE_CNT] AS
```

```
[ INTERFACE_COUNT ], [ NNM_OBJECTS ]. [ OVW_ID ] AS [ NNM_ID ]  
CONDITION= [ NNM_NETWORKS ]. [ TOPO_ID ] = [ NNM_OBJECTS ]. [ TOPO_ID ]  
  
[ SEGMENT ]  
SOURCE= [ NNM_SEGMENTS ], [ NNM_OBJECTS ]  
PARENT=NETWORK  
PARENT_RELATION= [ NNM_SEGMENTS ]. [ NET_ID ] = [ NNM_ID ]  
ATT= [ NAME ], [ NNM_ID ]  
COLUMNS= [ NNM_SEGMENTS ]. [ SEGMENT_NAME ] AS  
[ NAME ], [ NNM_SEGMENTS ]. [ NET_ID ] as  
[ NET_ID ], [ NNM_OBJECTS ]. [ OVW_ID ] AS [ NNM_ID ]  
CONDITION= [ NNM_SEGMENTS ]. [ TOPO_ID ] = [ NNM_OBJECTS ]. [ TOPO_ID ]
```

**Step 2.** Enter the data source (DSN) information.

1. Enter the name of the data source you will export data from. For this example nnm6 is being used as the data source. You must use this same name when setting up your ODBC connection.
2. Enter the name this integration can use to login to that data source.
3. Enter the password that goes with the login name.

**Step 3.** Specify the system settings.

1. The first four headers are used to specify if a log file will be created (LOG=TRUE), if the output file is of the XML type (XML=TRUE) or text (TXT=FALSE) and if a dump file should be created (DUMP=TRUE). In this example all files will be created except a TXT file.
2. Enter the locations that you want the LOG FILE, OUTPUT\_FILE and XML\_OUTPUT\_FILE placed after creation.
3. Enter the APPLICATION\_NAME for the application you will be exporting data from. For this example the data source application is NNM6.
4. In the Encoding line, enter the language type you will be viewing the configuration file in. ISO-8859-1 is entered in the example so that it can be viewed with a European language character set. There are numerous other character sets, for example Kanji uses completely different characters.

**Step 4.** Define the classes that will be exported.

### Using an Example Configuration File

1. Enter the names of all classes you will export. You can decide what the class names are, these class names will be used later with the import mapping. In the example the class names are NETWORK, and SEGMENT.
2. Enter a CLASS section for each class you will export.
3. Under each class section enter the SOURCE database the class data needs to be exported from.
4. For each class you must define attributes in the ATT section. The ATT section is used to identify the item and defines where in Service Desk the data must go. The attributes can literally be the same as the columns or aliases. The attributes will be captured from the columns that are written in the COLUMNS section.
5. Enter the COLUMNS you want to select with an alias [columnname1] AS [alias 1]. Columns that appear in the PARENT\_RELATION have to be put into the column list when tables are cached [LOADTABLE=TRUE], because the extractor uses this information to find all children for a particular parent in memory. If an alias is not specified the column name will be used as the alias name.
6. Enter the CONDITION or selection criteria for the records to be retrieved.

For additional information on configuring the extractor, see the section “Configuring the Extractor” on page 37.

## Importing From an MS Excel Spreadsheet

This example demonstrates how to import data from an MS Excel spreadsheet.

**Step 1.** Adjust the Excel spreadsheet

1. Add a heading record.
2. Select all data from the Excel sheet.
3. Enter a name for the selection in the upper left corner. The data selected for this example was given the name; PHONELBL.

**Figure D-1 Adjust the Excel Spreadsheet**

	A	B	C	D
1	LASTNAME	FIRSTNAME	JOBTITLE	SEARCHCODE
2	Galster	Steven	Director of OEM & ISV Sales	SGALSTER
3	Hinman	Dave	VP Business Development	DHINMAN
4	Myers	Stephen	Senior Sales Engineer	SMYERS
5	Barrea	Donna	Sales Administrator	DBARREA
6	Bentz	Ray	Director of ATG	BENTZ
7	Bergenholtz	Erik	ATG Engineer	BERGEN
8	Bergman	Barry	Sr. Director of Sales	BBERGMAN
9	Booth	David	Director of Training	BOOTH
10	Byrd	Megan	Inside Sales Rep-BD	MBYRD
11	Campana	Sal	ATG	SCAMPANA
12	Dondero	Robert	Training	DONDERO
13	Druss	Byron	Worldwide Sales Manger, Enterprise Services	BYRON

4. Save the Excel spreadsheet. The example is saved as `phone.xls`.

**Step 2.** Create an ODBC Data Source for the Excel spreadsheet

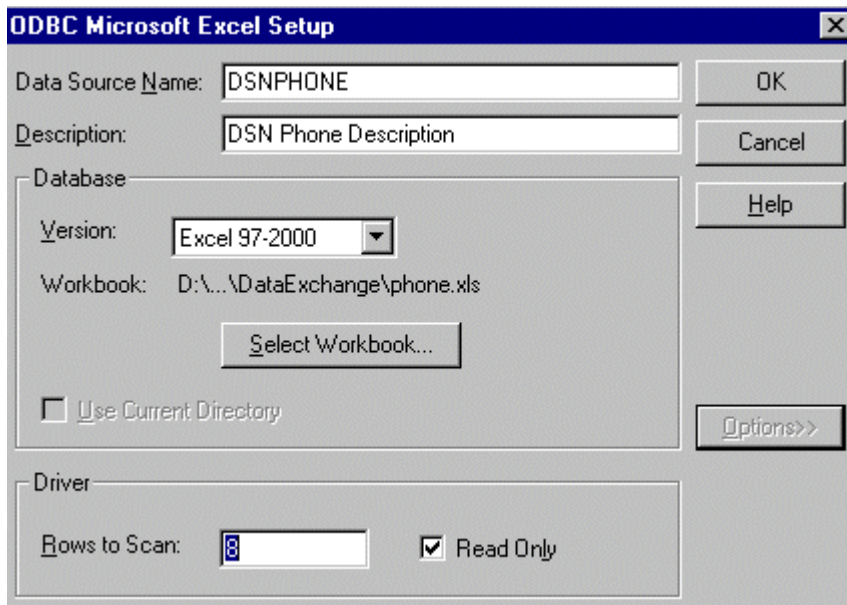
1. Start the ODBC Data Sources from the Windows Control Panel.
2. Select the `System DSN` tab and click `Add`.

## Examples

### Importing From an MS Excel Spreadsheet

3. Select the driver you want to use. **Microsoft Excel Driver (\*.xls) version 4.00.4202.00** was used for this example, then click **Finish**.
4. In the ODBC Microsoft Excel Setup dialog box enter a **Description**, then use the **Select Workbook** button to select the spreadsheet you selected in Step 1. Click **OK** and the **Data Source Name** field will automatically be filled. In this case it becomes **DSNPHONE**.

**Figure D-2 ODBC Microsoft Excel Setup dialog box**



#### **Step 3.** Configure the extractor.

1. Open one of the example configurable extractor provided with Service Desk, `sd_event.ini`.
2. Create a new ini file. In this example the new ini file is named `phoneconf.ini`.
3. Add the lines in the example so that they match your environment, and include the data you want to extract from the Excel file as follows:



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

[ SYSTEM ]
LOG=TRUE
XML=TRUE
TXT=FALSE
DUMP=TRUE
LOG_FILE=\phone.log
OUTPUT_FILE=\phone.xls
XML_OUTPUT_FILE=\phone.xml
APPLICATION_NAME=PHONELIST
ENCODING=ISO-8859-1

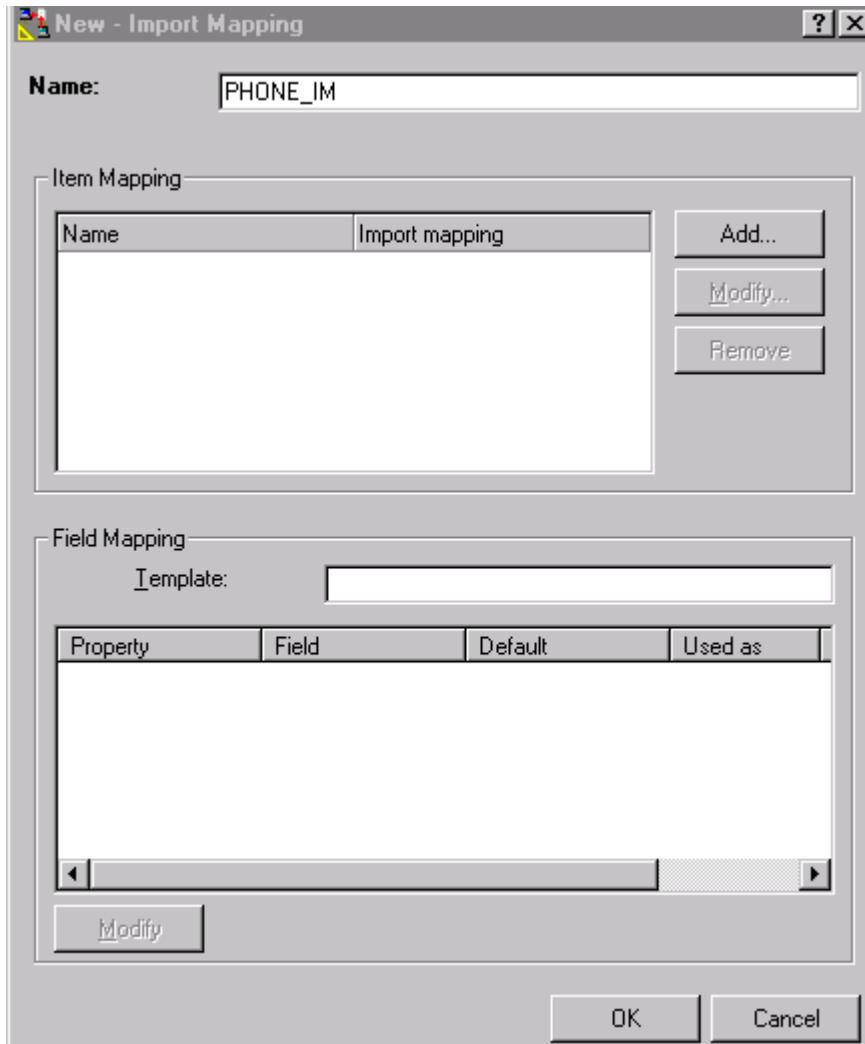
[ CLASSES ]
NAME=CLASSPHONE

[ CLASSPHONE ]
SOURCE=PHONELBL
ATT=[ LASTNAME ], [ FIRSTNAME ], [ JOBTITLE ], [ SEARCHCODE ]
COLUMNS=[ LASTNAME ]. [ FIRSTNAME ], [ JOBTITLE ], [ SEARCHCODE ]
LOADTABLE=TRUE
```

#### **Step 4. Create an import mapping.**

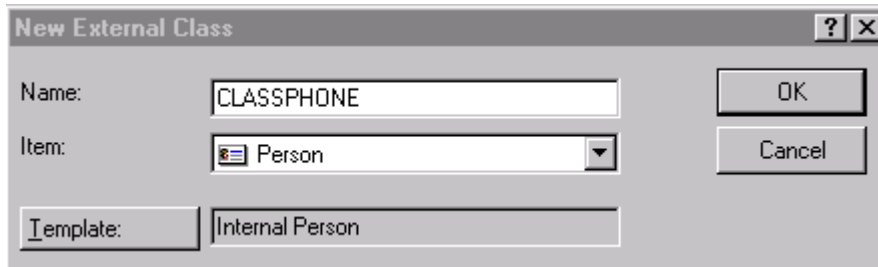
1. **From the Tools menu in Service Desk select System and then open the Data folder. Double-click Import Mapping. Right-click to create a new import mapping:**

**Figure D-3** New Import Mapping dialog box



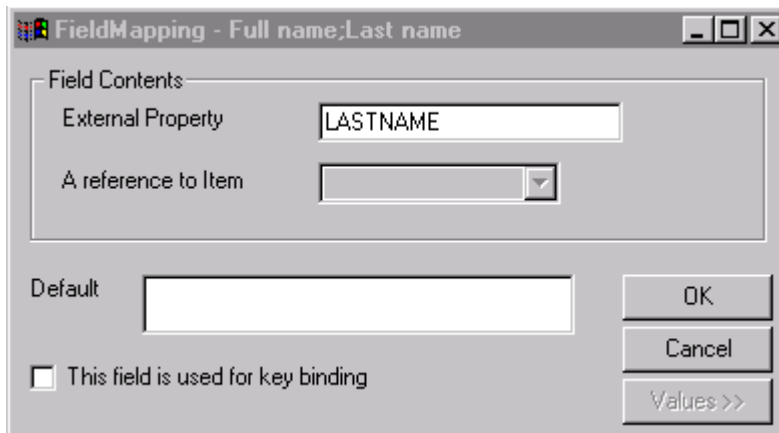
2. Give the import mapping a name, for this example it is `PHONE_IM`.
3. Create a New External Class for the Excel sheet. In this example it is `CLASSPHONE`. The mapping you create is dependent upon the type of information you are importing, because the example contains personnel information it is mapped to the `Person` item and the `Internal Person` template as follows:

**Figure D-4** New External Class dialog box



4. Click OK. The attributes assigned to the template and item selected for the class will be shown in the lower portion of the New Import Mapping dialog box.
5. Double-click the attributes you want to map to rows and columns in the Excel spreadsheet. In the Field Mapping dialog box you can enter the External Property name that corresponds to the Service Desk attribute you selected:

**Figure D-5** Mapping Attributes



---

**NOTE**

You must have at least one unique attribute designated as the binding key, this is usually the search code or another unique key.

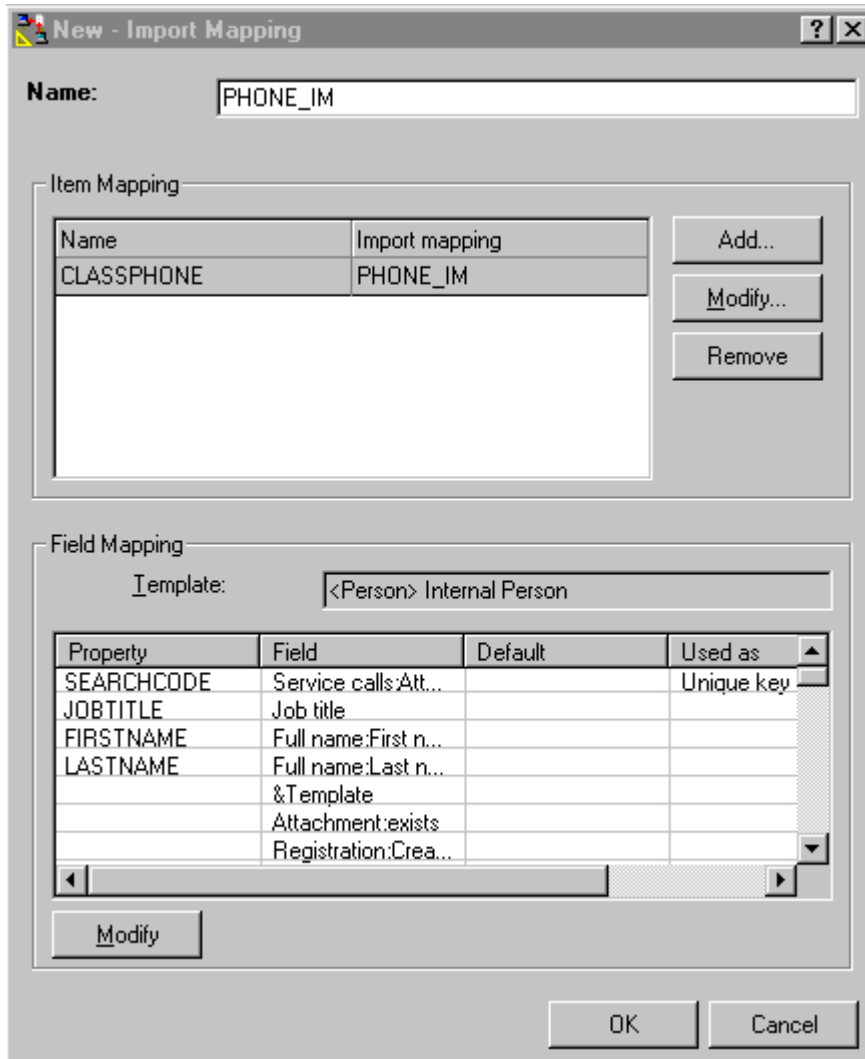
---

Examples

## Importing From an MS Excel Spreadsheet

6. Click **OK** to save your mapping and return to the Import Mapping dialog box. The following dialog box shows the completed import mapping for this example:

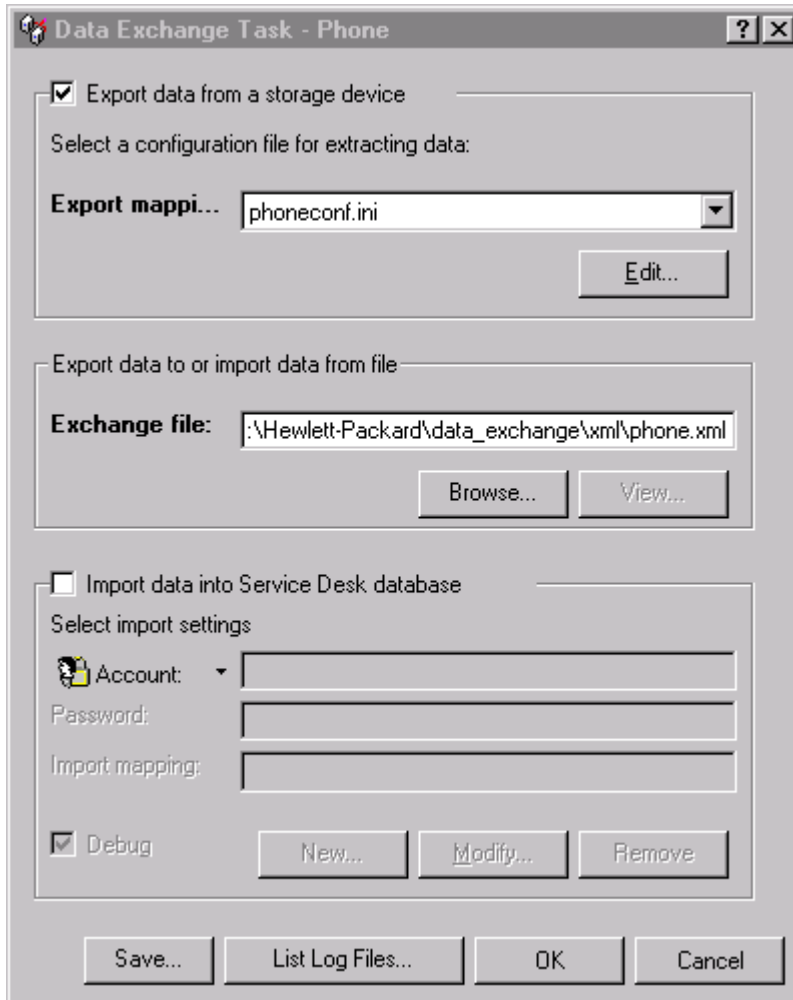
**Figure D-6 Complete Import Mapping**



**Step 5.** Export the Excel spreadsheet data.

1. Right-click in the Data Exchange Task window to create a New Data Exchange Task. An empty Data Exchange dialog box will open.
2. Select the Export Data from a storage device check box.
3. Enter the name of the ini file you created in the Export mapping field. For this example it is `phoneconf.ini`.
4. In the Exchange file field, enter the name you specified for it when configuring the ini file. For this example the file will be called `phone.xml`:

**Figure D-7**      **Export Data**



5. Click **Save** to save this task for use another time, and then **OK** to export the data from the Excel file.
6. Check the `phoneconf_exp.log` file to verify that the export was successful.

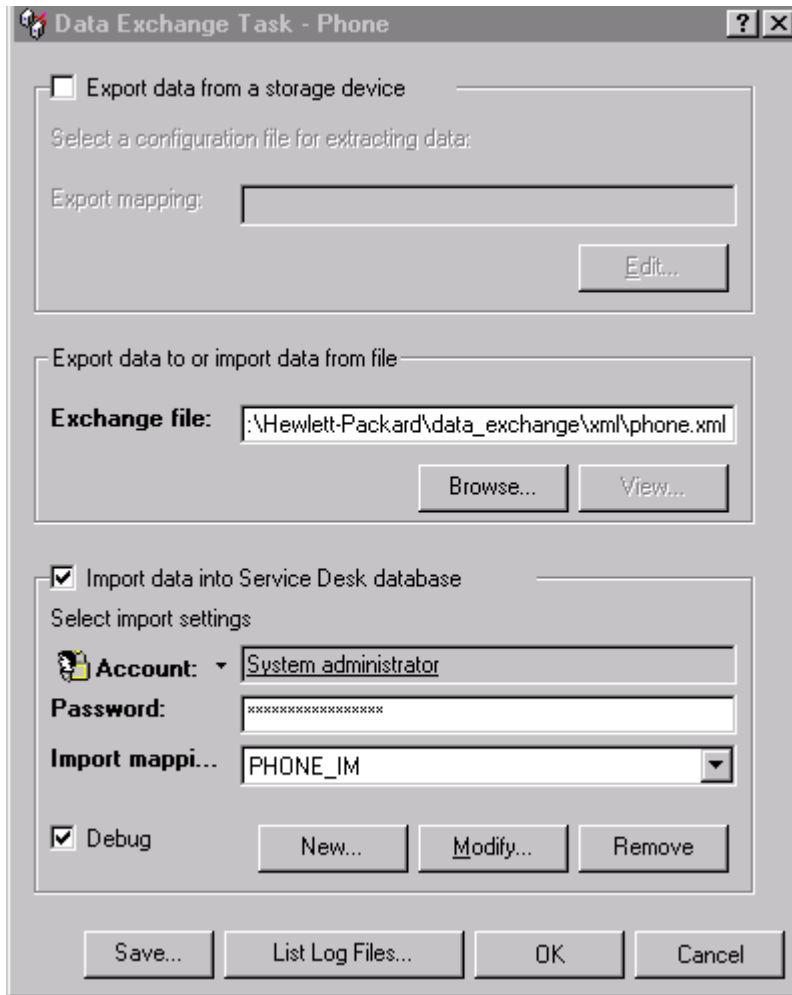
**Step 6.** Import the Excel data.

1. Right-click in the Data Exchange Task window to create a New Data

Exchange Task. An empty Data Exchange dialog box will open.

2. In the Exchange file field, enter the name of the phone.xml file.
3. Select the Import data into Service Desk database check box.
4. Enter a Service Desk account and password created for integration purposes.
5. In the Import mapping field use the drop-down arrow to find the import mapping you created for importing the Excel file. In this example it is **PHONE\_IM**:

**Figure D-8**      **Import Data**



6. Select the Debug check box and click OK to import the data.
7. Check PHONE\_IM\_imp.log to verify that the import was successful.



---

## Importing Data With Relations

This example focuses on the process of configuring the extractor file and creating the import mapping to import data from a data source. Special attention is given in this example to how you can 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 you will export and how it is stored.
2. Create an ODBC link
3. Configure the Extractor
4. Export data
5. Map the exported data to templates, items and fields in Service Desk.
6. Import the data.

### Data Source

To configure the extractor (ini file) 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 a Data Dictionary is provided with this type of information. The Service Desk Dictionary is useful when mapping where in Service Desk the data will be imported to and the relations between items in Service Desk.

The data used for this example was 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 “Defining the ODBC Link” on page 196, and the task of defining areas in your excel file, “Preparing the Excel Sheet for Data Exchange” on page 195. When looking at your data you should always think in terms of the SOURCE or database table, and COLUMNS within the tables. In the Excel spreadsheet the information is also grouped into tables, SOURCES, and columns, COLUMNS.

**Figure D-9**                    **EMPLOYEE SOURCE**

A	B	C	D
<b>EMPLOYEE</b>			
SEARCHCODE	FIRSTNAME	LASTNAME	EMAIL
EMP100003	Maarten	van Dijk	maarten_vandijk@acme.com
EMP100004	John	Smith	john_smith@acme.com
EMP100005	Shelley	Long	shelley_long@acme.com
EMP100006	Guy	François	guy_francois@acme.com
EMP100007	Helen	Morris	helen_morris@acme.com
EMP100008	Monica	van Velden	monica_vanvelden@acme.com
EMP100009	Frank	Zappa	frank_zappa@acme.com
EMP100010	Joe	Barbarese	joe_barbarese@acme.com
EMP100011	Pete	McPherson	pete_mcpherson@acme.com
EMP100012	Sandra	Berke	sandra_berke@acme.com
EMP100013	Michael	Sasek	michael_sasek@acme.com
EMP100014	Zem	Philips	zem_philips@acme.com

**Figure D-10**                    **EMPLOYEE SOURCE - Continued**

E	F	G	H
TELEPHONE	NUMBER	ORG	WG
303	303	HPHOLLAND	ORGHELP
304	304	HPFRANCE	ORGPC
305	305	HPHOLLAND	ORGPC
306	306	HPSPAIN	ORGHELP
307	307	HPUK	ORGIT
308	308	HPFRANCE	ORGPC
309	309	HPGERMANY	ORGIT
310	310	HPHOLLAND	ORGIT
311	311	HPSPAIN	ORGHELP
312	312	HPGERMANY	ORGHELP
313	313	HPSPAIN	ORGIT
314	314	HPSPAIN	ORGIT

**Figure D-11 ORGANIZATION, SOFTWARE & WORKGROUP SOURCES**

ORGANISATION		SOFTWARE	
SEARCHCODE2	NAME	SEARCHCODE3	NAME3
ORG20001	HPHOLLAND	NT4	NT4.0
ORG20002	HPFRANCE	W2000	WIN2000
ORG20003	HPUK		
ORGANISATION		WORKGROUP	
SEARCHCODE4	NAME4	SEARCHCODE4	NAME4
ORG20004	HPGERMANY	ORGHELP	ORG-Helpdesk
ORG20005	HPSPAIN	ORGPC	ORG-PC
		ORGIT	ORG-IT

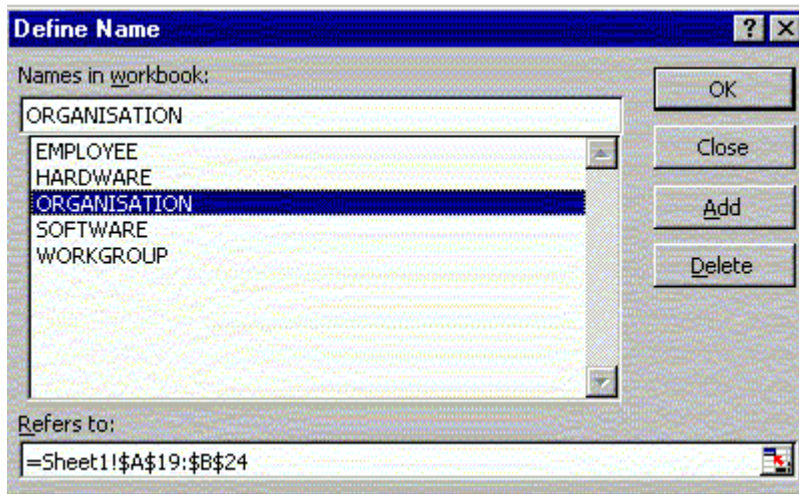
**Figure D-12 HARDWARE SOURCE**

HARDWARE			
SEARCHCODE5	NAME5	SW	USER
ORGPC001	ORGPC1	NT4.0	EMP100005
ORGPC002	ORGPC2	WIN2000	EMP100009
ORGPC003	ORGPC3	WIN2000	EMP100006
ORGPC004	ORGPC4	NT4.0	EMP100014
ORGPC005	ORGPC5	NT4.0	EMP100012
ORGPC006	ORGPC6	WIN2000	EMP100012
ORGPC007	ORGPC7	WIN2000	EMP100008

### Preparing the Excel Sheet for Data Exchange

When exporting data from an Excel sheet you will need to define the areas of data in your excel sheet. In essence you will be grouping the data into tables that are easier to identify for import mapping. In this example the following areas are defined in the Excel sheet; EMPLOYEE, ORGANISATION, SOFTWARE, WORKGROUP, and HARDWARE. To define an area in an Excel sheet; from the File menu select Insert, Name, and then Define. You can then set or view the definition for each area as shown in the following example:

**Figure D-13**      **Excel File Name Definition**



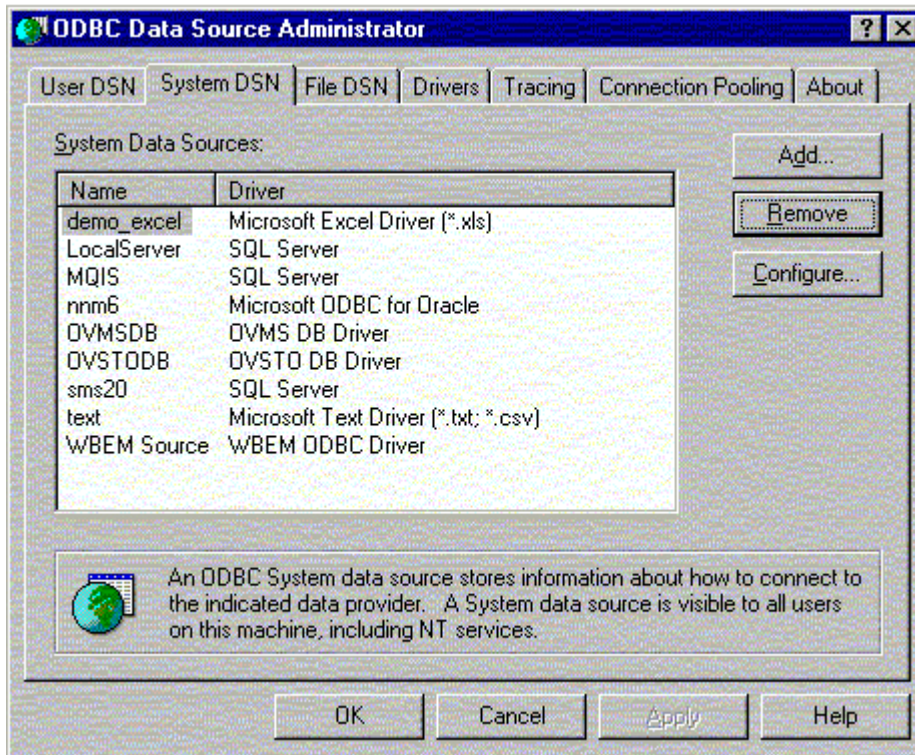
After you create the ODBC link and create or modify the configuration file you can export the data using the Data Exchange dialog box. Clear the option for importing data until after you have had time to verify that your export was successful. For more information see “Using a Task to Export Data From a Storage Device” on page 51.

### **Defining the ODBC Link**

An ODBC connection must be configured on the application server where the data you will import is located. For example, if the exported XML file is on your Service Desk application 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 ODBC links see “Defining the ODBC link” on page 46.

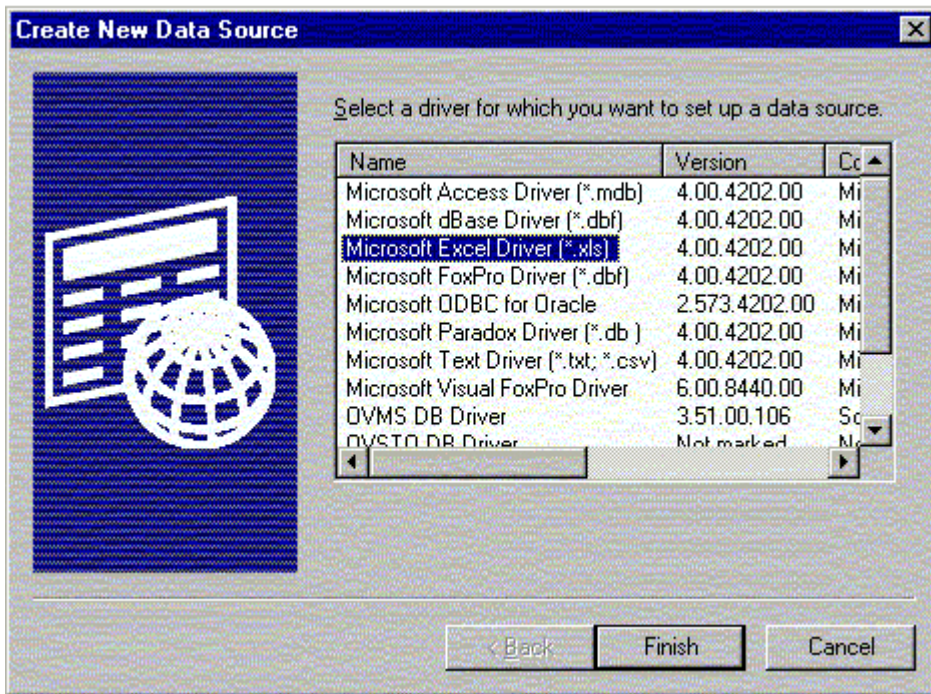
1. From the System DSN tab select Add to create a new ODBC link:

**Figure D-14 Create New ODBC Connection**



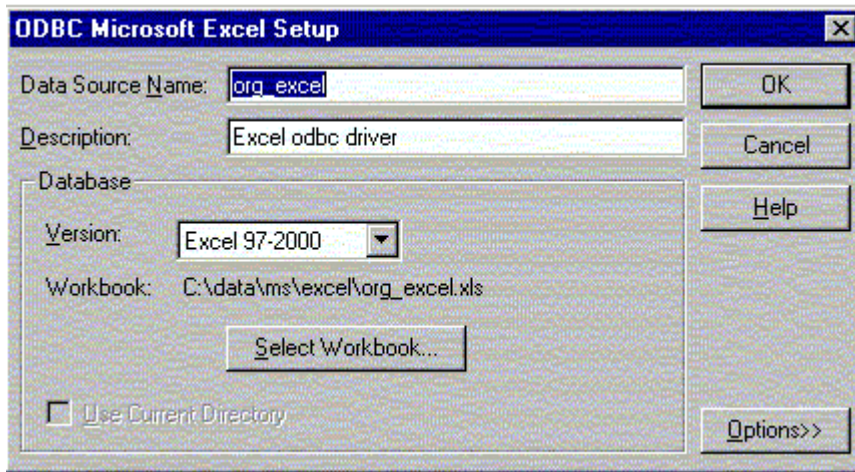
2. Select the Microsoft Excel Driver, it will have an \*.xls extension, then click Finish:

**Figure D-15** New Data Source



3. Type the Data Source Name and a Description. The Data Source Name should match the DSN name in your configuration file, the description is for your own reference.
4. Click Select Workbook to locate the data source that you will use, then click OK to finish the procedure:

**Figure D-16 Excel ODBC\_DSN**



## Configuring the Extractor and the Import Mapping

This section explains how the example `org_excel.ini` configuration file was created and the special attention given to exporting relations. The extractor is an executable extractor file that uses parameters to connect to the external source and extract data. After the data is exported the import mapping needs to be configured correctly for importing the relations.

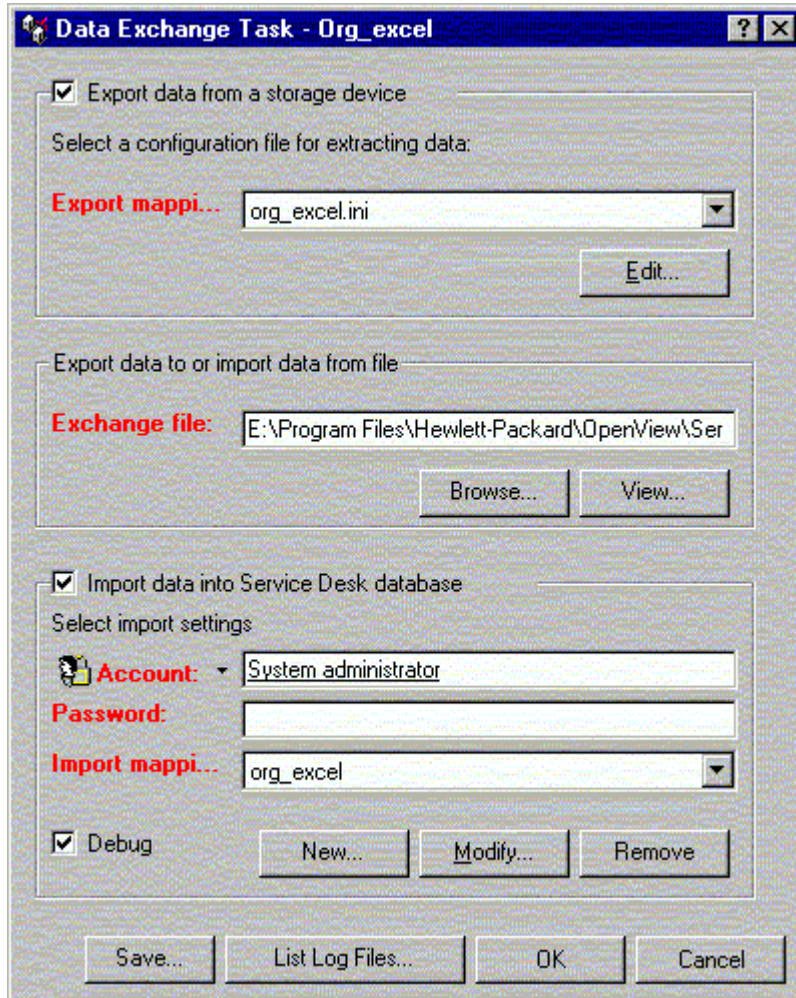
Mapping data from the XML file to Service Desk items and fields can be a labor intensive job, if you cannot use one of the import mapping examples provided you can make the process easier by creating a well structured configuration file and making sure it works before going on to the import mapping. If you don't you might find yourself creating the import mapping more than once. For this example the import mapping is shown after each class in the configuration file is described, to better demonstrate how what is entered in the configuration file effects what needs to be entered in the import mapping. Normally you would create the entire configuration file and then make the import mapping based on the information in the XML file.

The export process can be started from the application user interface or from the command line, see "Exporting Data" on page 51, or "Exporting From the Command Line" on page 53 for additional information. The configuration file can be opened from the Data Exchange dialog box, click

Examples  
Importing Data With Relations

Edit to edit or view the configuration file selected in the Export mapping field as shown below:

**Figure D-17** Export Mapping in Data Exchange dialog box



In this example the following Classes will be configured for export by the



extractor:

<b>Class</b>	<b>Description</b>
SOFTWARE	This class is selected to export the different software items from the data source into the XML file.
ORG	This class will export the organization items from the data source into the XML file.
EMP	This class will export the employees and the employee attributes entered into the XML file: Email addresses, and employee organization.
PHONE	This class will export the employee telephone numbers from the data source to the XML file.
HARDWARE	This class will export the different hardware items from the data source into the XML file.
HWUSERS	This class will export the different hardware users from the data source into the XML file.
REL	This class will export the different relationships from the data source into the XML file.
WORKGROUP	This class will export the different workgroups from the data source into the XML file.
WORKGROUPEMP	This class will export the different workgroup members from the data source into the XML file.

### Example D-1

#### Configuration File

```
[DSN]  
NAME=org_excel  
USR=  
PWD=  
  
[SYSTEM]  
LOG=TRUE  
XML=TRUE  
TXT=FALSE  
DUMP=TRUE
```

## Examples

### Importing Data With Relations

```
LOG_FILE=org_excel.log
OUTPUT_FILE=org_excel.txt
XML_OUTPUT_FILE=org_excel.xml
APPLICATION_NAME=org_excel
```

```
[ CLASSES ]
```

```
NAME=SOFTWARE , ORG , EMP , PHONE , HARDWARE , HWUSERS , REL , WORKGROUP , WORKGROUPEMP
```

```
[ SOFTWARE ]
```

```
SOURCE=SOFTWARE
```

```
ATT=[ SEARCHCODE3 ] , [ NAME3 ]
```

```
COLUMNS=[ SEARCHCODE3 ] , [ NAME3 ]
```

```
LOADTABLE=TRUE
```

```
[ ORG ]
```

```
SOURCE=ORGANISATION
```

```
ATT=[ SEARCHCODE2 ] , [ NAME ]
```

```
COLUMNS=[ SEARCHCODE2 ] , [ NAME ]
```

```
LOADTABLE=TRUE
```

```
[ EMP ]
```

```
SOURCE=EMPLOYEE
```

```
ATT=[ SEARCHCODE ] , [ FIRSTNAME ] , [ LASTNAME ] , [ EMAIL ] , [ ORG ]
```

```
COLUMNS=[ SEARCHCODE ] , [ FIRSTNAME ] , [ LASTNAME ] , [ EMAIL ] , [ ORG ]
```

```
LOADTABLE=TRUE
```

```
[ PHONE ]
```

```
SOURCE=EMPLOYEE
```

```
ATT=[ SEARCHCODE ] , [ PHONENUMBER ]
```

```
COLUMNS=[ SEARCHCODE ] , [ NUMBER ] AS [ PHONENUMBER ]
```

```
LOADTABLE=TRUE
```

```
[ HARDWARE ]
```

```
SOURCE=HARDWARE
```

```
ATT=[ SEARCHCODE5 ] , [ NAME5 ]
```

```
COLUMNS=[ SEARCHCODE5 ] , [ NAME5 ]
```

```
LOADTABLE=TRUE
```

```
[ HWUSERS ]
```

```
SOURCE=HARDWARE
```

```
ATT=[ SEARCHCODE5 ] , [ USER ]
```

```
COLUMNS=[ SEARCHCODE5 ] , [ USER ]
```

```
LOADTABLE=TRUE

[REL]
SOURCE=HARDWARE
ATT=[SEARCHCODE5],[SW]
COLUMNS=[SEARCHCODE5],[SW]
LOADTABLE=TRUE

[WORKGROUP]
SOURCE=WORKGROUP
ATT=[SEARCHCODE4],[NAME4]
COLUMNS=[SEARCHCODE4],[NAME4]
LOADTABLE=TRUE

[WORKGROUPEMP]
SOURCE=EMPLOYEE
ATT=[SEARCHCODE]
COLUMNS=[SEARCHCODE],[WG] AS [SEARCHCODE4]
LOADTABLE=TRUE
PARENT=WORKGROUP
PARENT_RELATION=[WG]=[SEARCHCODE4]
PARENT_RELATION_NAME=PARENT
```

### **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 running Data Exchange from the user interface you only need to fill in the LOG, XML, TXT, DUMP and APPLICATION NAME fields. The fields LOG\_FILE, OUTPUT\_FILE, and XML\_OUTPUT\_FILE only need to be filled in if you run Data Exchange 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 will take each class defined in the CLASSES section and make SQL statements.

Sources, columns and classes are selected and relationships made according to the definitions in the configuration file, see “The Extraction Process” on page 49 for additional information. When you later import the data the objects are imported in the order they were written. The order takes some thought when developing the configuration file for importing relations. If you look in the configuration file that follows you will see that WORKGROUP will be extracted before WORKGROUPEMP, because WORKGROUPEMP is the child relation of WORKGROUP.

---

**NOTE**

If a class is not in the CLASSES file it will not be imported, even if the class is described later in the configuration file. You will not get an error message 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 Service Desk.

---

### **Configuring the SOFTWARE Class**

In the data source the class called Software is stored in the table called SOFTWARE, from the excel spreadsheet it is a section defined as SOFTWARE in the Excel sheet. The table is the SOURCE in the configuration file. Within the SOURCE two attributes are selected for export, the attributes are put in the ATT field. In the COLUMNS field you need to specify where within the SOURCE those attributes, ATT items, are coming from. For this example class, the COLUMN names and ATT names match so no aliases are necessary.

You always need to include a SEARCHCODE. The SEARCHCODE is a unique key that must always be exported and mapped to the field `search code` in Service Desk for identifying the class. For Software the column SEARCHCODE3 contains the unique code name given to each software item, the NAME3 column contains the name the different software items were given.

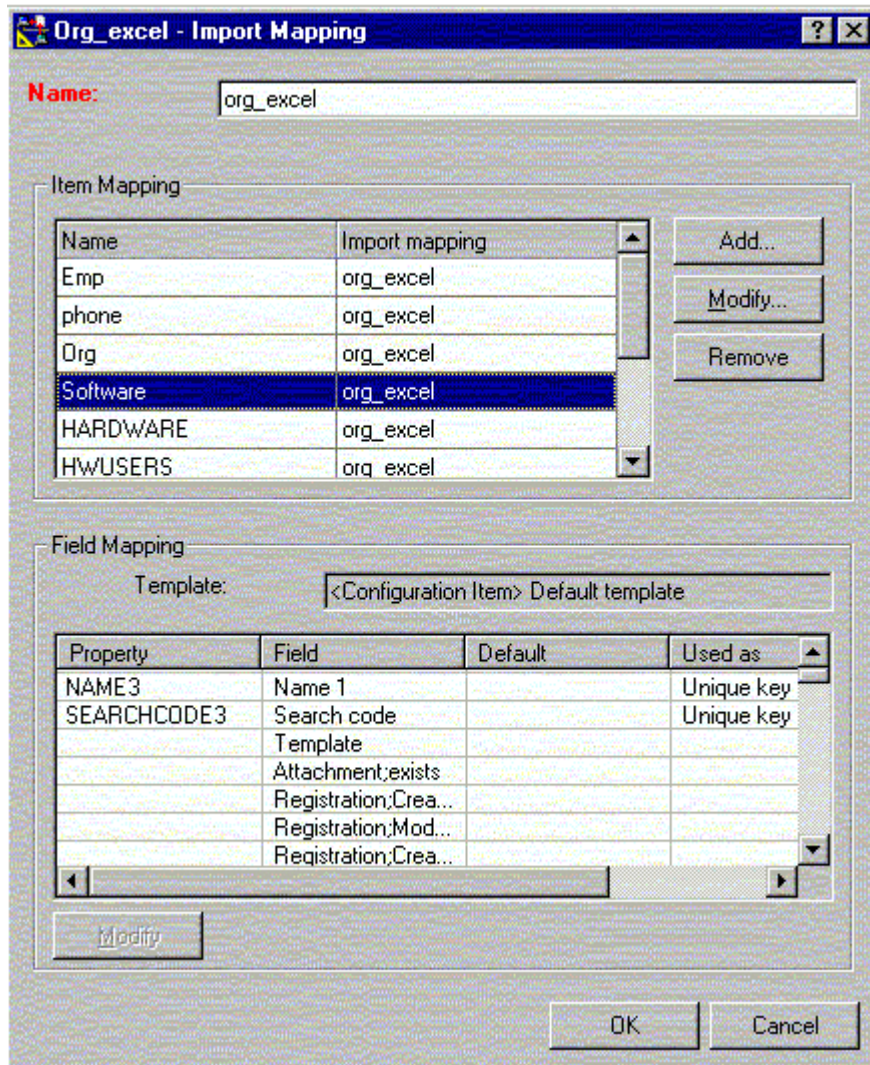
The LOADTABLE field is used to specify how the data is stored while it is being extracted. In this case the LOADTABLE is set to TRUE so that the records will be cached in memory, because the class in this example does not have any parent-child relations that need to be searched for.

### **Import Mapping for the SOFTWARE Class**

In the import mapping the class name in brackets [SOFTWARE] is mapped to the item called `Configuration Item` in Service Desk. The

default template is used for this class. The attributes in the ATT line are mapped to Fields that are available in the default template. In this case [SEARCHCODE3] is mapped to the Search code field and [NAME3] is mapped to the Name1 field. Both attributes are unique, the mapping can be extended later by modifying the configuration file to include more attributes. The completed import mapping is displayed in the following dialog box:

**Figure D-18 Import Mapping - SOFTWARE**



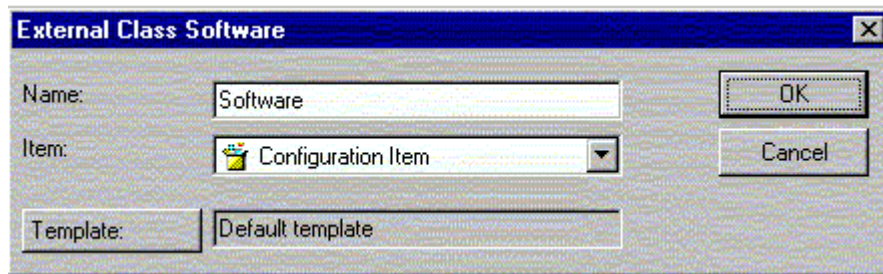
Scrolling down in the dialog box above will show the following additional fields:

Field	Default
Admin.Org	Invention Inc.

Field	Default
Max.Installations	1

The following dialog box shows the import mapping for the external class to the Configuration Item in Service Desk and the template selected:

**Figure D-19 Mapping External Class - SOFTWARE**



### Configuring the ORG Class

This class is imported in a similar manner as the class SOFTWARE. One major difference is that the SOURCE for the class has a different name. The class name is ORG and the SOURCE is the ORGANISATION table. This shows that you can give a class any name you want in the configuration file as long as the SOURCE is provided. Keep in mind that when the exported data is put into the XML file it will be sorted under the class name that you give it and that class name is what you will need to map to an item in Service Desk.

Another point to note about this class is that this class is intentionally put in the configuration file before the class EMP. The reason for that is because EMP will export ORG as an attribute.

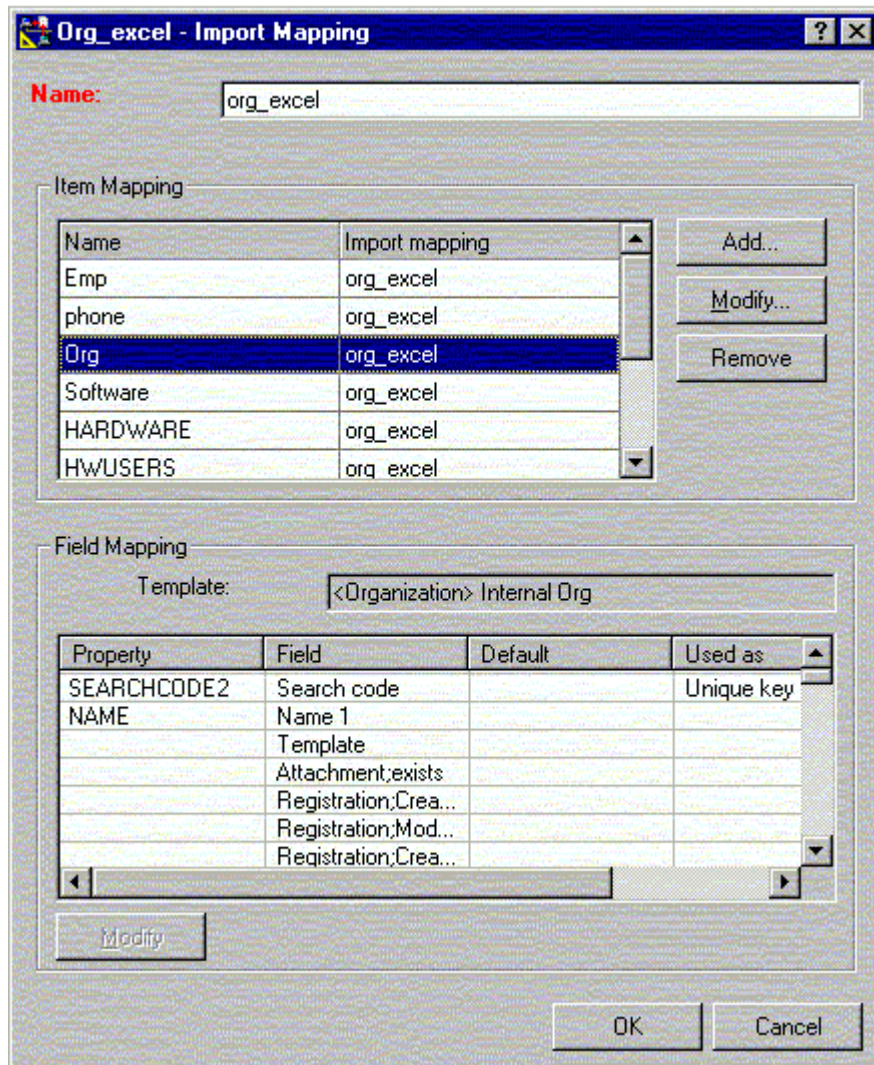
### Import Mapping for the ORG Class

In the import mapping the class name in brackets [ORG] is mapped to the item called Organization in Service Desk. The template called Internal Org is used for this class. The attributes in the ATT line are mapped to Fields that are available in the Internal Org template. In this case [SEARCHCODE2] is mapped to the Search code field and [NAME] is mapped to the Name1 field. The mapping can be extended later by modifying the configuration file to include more attributes. The

Examples  
Importing Data With Relations

completed import mapping is displayed in the following dialog box:

**Figure D-20 Import Mapping - ORG**



Scrolling down in the dialog box above will show the following additional

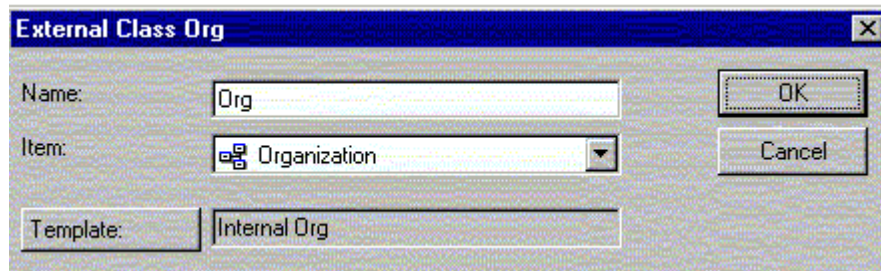


fields:

Field	Default
Status	Active
Category	Organization

The following dialog box shows the how the external class was mapped to the `Organization` item in Service Desk and the template selected:

**Figure D-21 Mapping External Class - ORG**



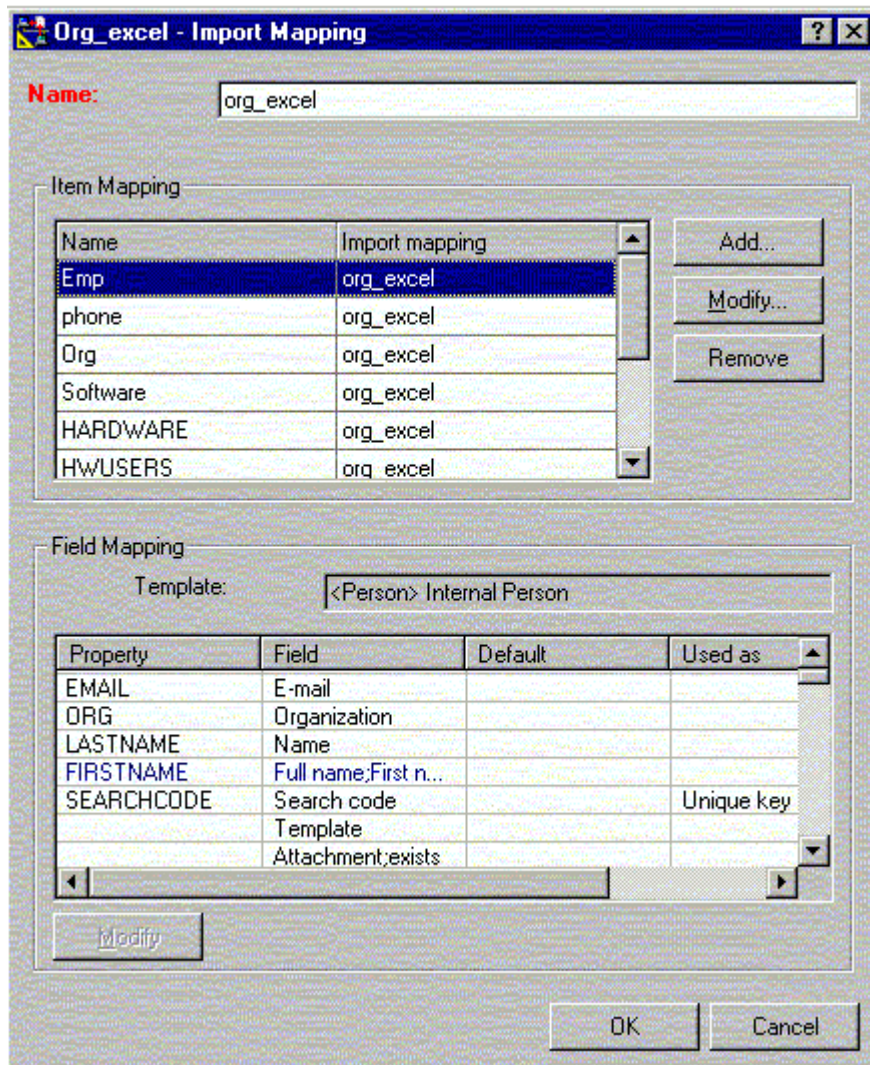
### Configuring the EMP Class

The class EMP is configured to be imported before the following class, PHONE. That is because phone numbers will be linked to employees.

### Import Mapping for the EMP Class

In the import mapping the class name in brackets [EMP] is mapped to the item called Person in Service Desk. The default template is used for this class. The attributes in the ATT line are mapped to Fields that are available in the default template. In this case [SEARCHCODE] is mapped to the Search code field, [ORG] is mapped to the Organization field, [LASTNAME] is mapped to Name, [FIRSTNAME] is mapped to Full name; First name, and [EMAIL] is mapped to E-mail. Search code is the unique key for identifying the class. The completed import mapping is displayed in the following dialog box:

**Figure D-22 Import Mapping - EMP**



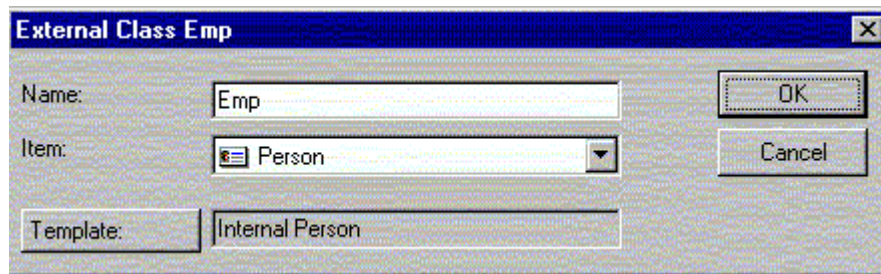
Scrolling down in the dialog box above will show the following additional fields:

Field	Default
Status	Active

Field	Default
Category	Employee

The following dialog box shows the how the external class was mapped to the Person item in Service Desk and the template selected, in this case the Internal Person template:

**Figure D-23 Mapping External Class - EMP**



You will need to create an item reference for the ORG property. To create the item reference:

1. Double-click the `organization` field, in the Import Mapping dialog box, to open the field mapping.
2. In the Field Mapping dialog box the External Property field should contain the ORG property.
3. In the A reference to Item field, use the drop-down arrow to enter the Name1 item.
4. Click OK to finish.

### Configuring the PHONE Class

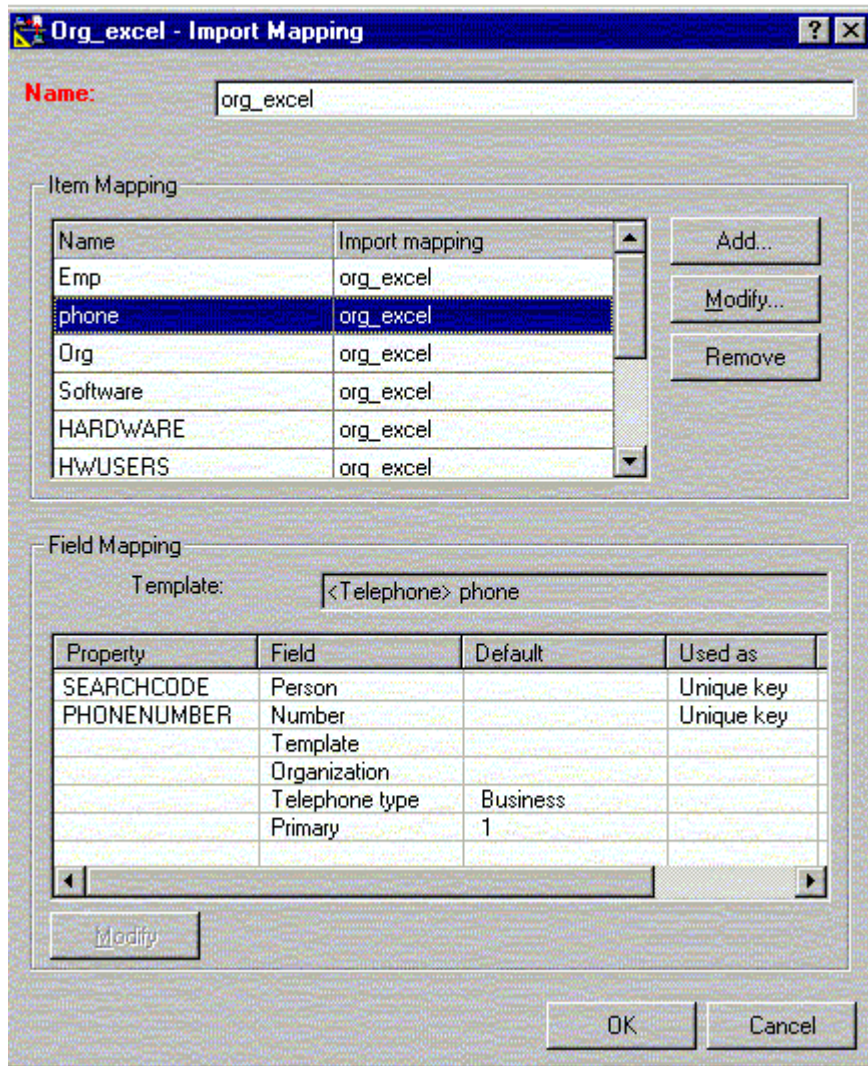
The PHONE class is taken from the EMPLOYEE SOURCE. This class is being exported as a separate class because later it will be imported to a different item with a different template. The EMP class will be imported to the Person item and the PHONE class will be imported to the Telephone item in Service Desk. When looking at the ATT and COLUMN names you will notice that they are different. ATT, attribute names are decided by you, just like class names. If they are different than the COLUMN names, if they have an alias, you need to include that with the AS statement in the COLUMN line. In the example the ATT

called PHONENUMBER will be exported from the COLUMN called NUMBER and listed in the XML file as PHONENUMBER.

### **Import Mapping for the PHONE Class**

In the import mapping the class name in brackets [PHONE] is mapped to the item called Telephone in Service Desk. The phone template is used for this class. The attributes in the ATT line are mapped to fields that are available in the default template. In this case [SEARCHCODE] is mapped to the Person field and [PHONENUMBER] is mapped to the Number field. Both attributes are unique keys. Two default settings are included so that the number will be imported as the primary Business phone for that person. For information on importing multiple phone numbers, see the following example “Importing Multiple Telephone Numbers” on page 226. The import mapping for the PHONE class is displayed in the following dialog box:

**Figure D-24**      **Import Mapping - PHONE**



The following dialog box shows the how the external class was mapped to the Telephone item in Service Desk and the template selected:

**Figure D-25 Mapping External Class - PHONE**



You will need to create an item reference for the SEARCHCODE property. To create the item reference:

1. Double-click the Person field, in the import mapping dialog box, to open the field mapping dialog box.
2. In the Field Mapping dialog box the External Property field should contain the SEARCHCODE property.
3. In the A reference to Item field, use the drop-down arrow to enter the Search code item.
4. Click OK to finish.

### **Configuring the HARDWARE Class**

The HARDWARE class is intentionally listed for extraction before the class HWUSERS because it is the parent of that class.

---

#### **NOTE**

The classes; HWUSERS and WORKGROUPEMP are configured and imported differently in this example, yet the end result is the same. This demonstrates that there are multiple ways of configuring how items are imported into Service Desk. When configuring how relations will be imported into Service Desk it can be specified in the configuration file or in the import mapping. The process or doing it in the configuration file is simpler and provides better performance.

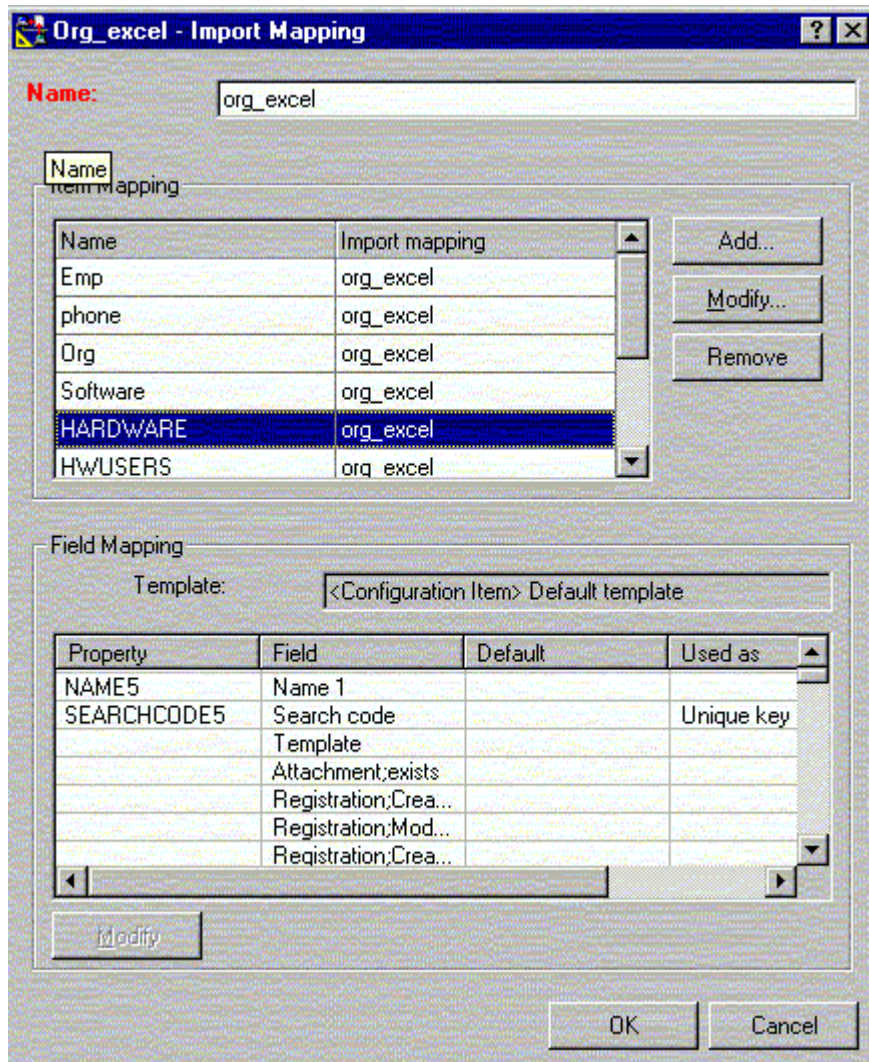
---

### **Import Mapping for the HARDWARE Class**

In the import mapping the class name in brackets [HARDWARE] is mapped to the item called Configuration Item in Service Desk. The

default template is used for this class. The attributes in the ATT line are mapped to fields that are available in the default template. In this case [SEARCHCODE5] is mapped to the Search code field and [NAME5] is mapped to the Name1 field. The completed import mapping is displayed in the following dialog box:

**Figure D-26 Import Mapping - HARDWARE**



Examples  
Importing Data With Relations

Scrolling down in the dialog box above will show the following additional fields:

Field	Default
Admin.Org	Invention Inc
Max.Installations	1
Unique	1
Status	Production

The following dialog box shows the how the external class was mapped to the Configuration Item in Service Desk and the template selected:

**Figure D-27 Mapping External Class - HARDWARE**



### Configuring the HWUSERS Class

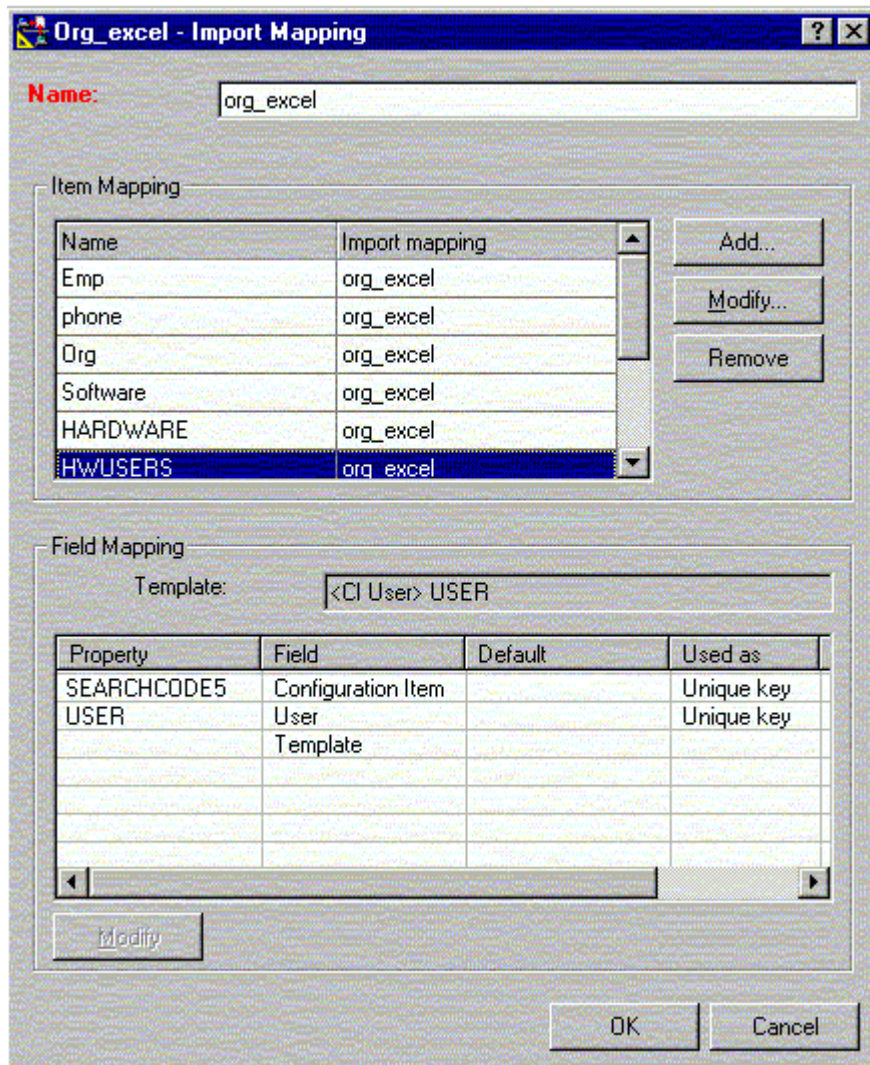
The class HWUSERS is the child class of HARDWARE and was created separately so that it can be imported to a different item in Service Desk. HARDWARE is imported as a configuration item and HWUSERS is imported as CI User in Service Desk.

### Import Mapping for the HWUSERS Class

In the import mapping the class name in brackets [HWUSERS] is mapped to the item called CI User in Service Desk. The USER template is used for this class. The attributes in the ATT line are mapped to fields that are available in the USER template. In this case [SEARCHCODE5] is mapped to the Configuration item field, because this is a child of that item. The [USER] attribute is mapped to the User field. The completed import mapping is displayed in the following dialog box:

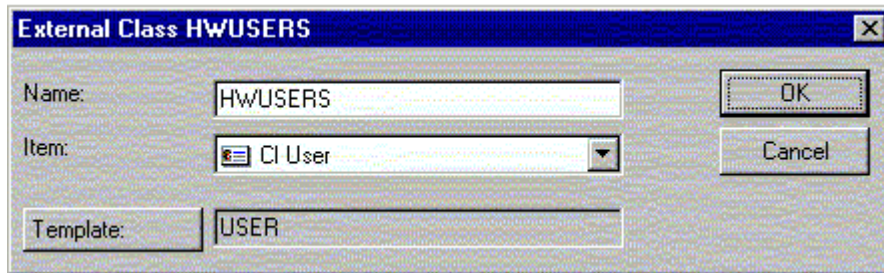


**Figure D-28**      **Import Mapping - HWUSERS**



The following dialog box shows the how the external class was mapped to CI User in Service Desk and the template selected:

**Figure D-29 Mapping External Class - HWUSERS**



You will need to create an item reference for the `SEARCHCODE5` and the `USER` properties. To create the item references:

1. Double-click the `Configuration Item` field, in the import mapping dialog box, to open the field mapping dialog box.
2. In the `Field Mapping` dialog box the `External Property` field should contain the `SEARCHCODE5` property.
3. In the `A reference to Item` field, use the drop-down arrow to enter the `Search code` item.
4. Click `OK` to return to the `Import Mapping` dialog box.
5. Double-click the `user` field to open the field mapping dialog box.
6. In the `Field Mapping` dialog box the `External Property` field should contain the `USER` property.
7. In the `A reference to Item` field, use the drop-down arrow to enter the `Search code` item.
8. Click `OK` to finish.

### Configuring the REL Class

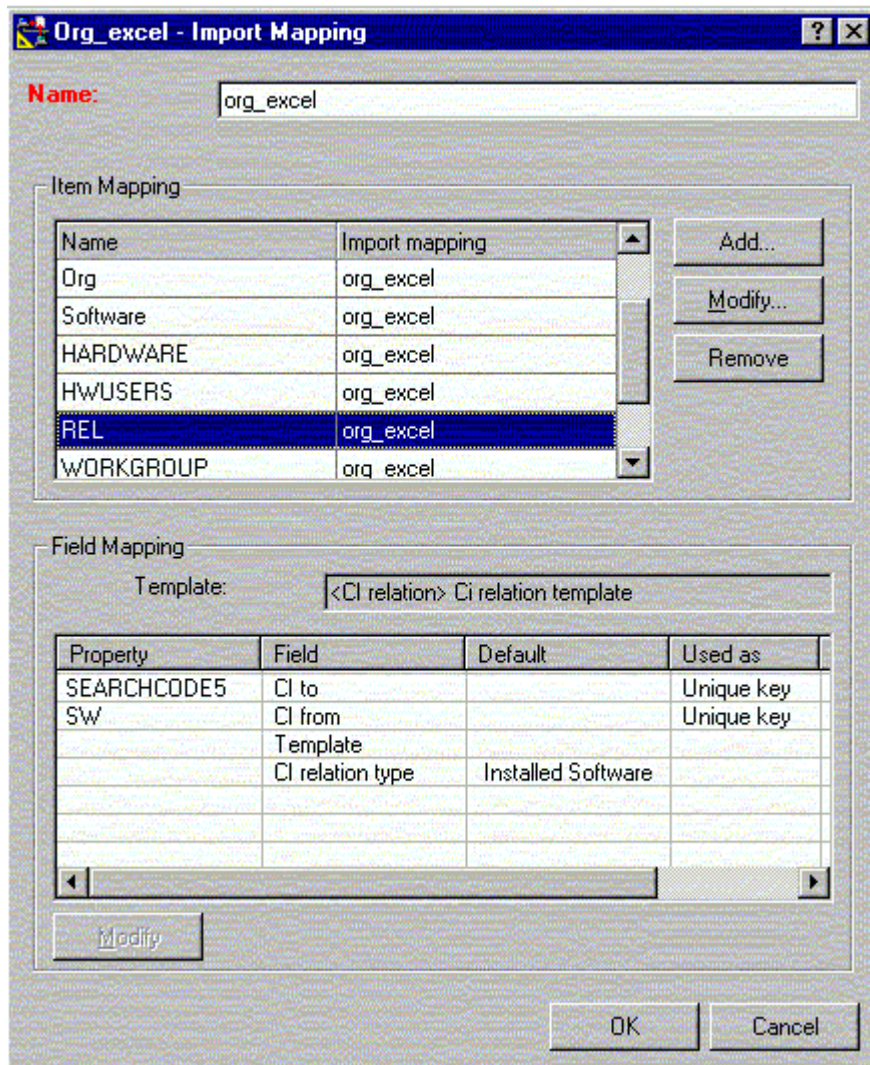
This class is being used to define the relations in the `SOURCE` called `HARDWARE` for the attributes listed. This is a `1:Rel:1` relation. The Class `REL` is mapped to the `CI Relation` item that contains a number of different relations as fields. In this manner you can specify the relation of the classes that are linked to this class.

### Import Mapping for the REL Class

In the import mapping the class name in brackets `[REL]` is mapped to the item called `CI Relation` in `Service Desk`. The `CI Relation` template

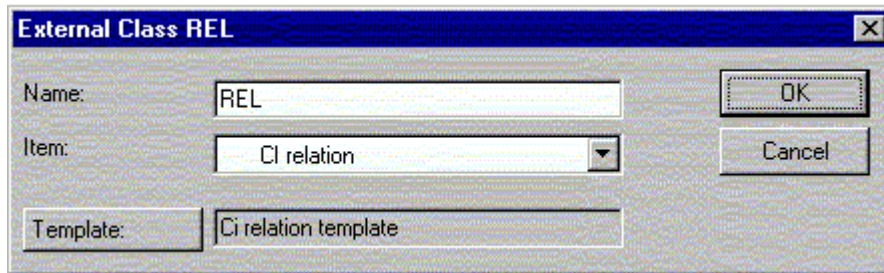
is used for this class. The attributes in the ATT line are mapped to fields that are available in the CI relation template. In this case [SEARCHCODE5] is mapped to the CI to field, and the attribute SW, which refers to software, is mapped to CI from. Any default values entered will be overwritten by the value mapping. You can enter a default value to define the relationship, for example if you look at the field CI relation type you will see that it gives this class the default value Installed Software. It is also possible to create a value mapping to define the relations or to reference the relation table in Service Desk. The completed import mapping is displayed in the following dialog box:

**Figure D-30**      **Import Mapping - REL**



The following dialog box shows how the external class is mapped to the CI Relation item in Service Desk and the template selected:

**Figure D-31 Mapping External Class - REL**



You will need to create an item reference for the SEARCHCODE5 and the SW properties. To create the item references:

1. Double-click the CI field, in the import mapping dialog box, to open the field mapping dialog box.
2. In the Field Mapping dialog box the External Property field should contain the SEARCHCODE5 property.
3. In the A reference to Item field, use the drop-down arrow to enter the Search code item.
4. Click OK to return to the Import Mapping dialog box.
5. Double-click the CI from field to open the field mapping dialog box.
6. In the Field Mapping dialog box the External Property field should contain the SW property.
7. In the A reference to Item field, use the drop-down arrow to enter the Name1 item.
8. Click OK to finish.

### **Configuring the WORKGROUP Class**

The class WORKGROUP is imported before WORKGROUPEMP because it is the parent of that class.

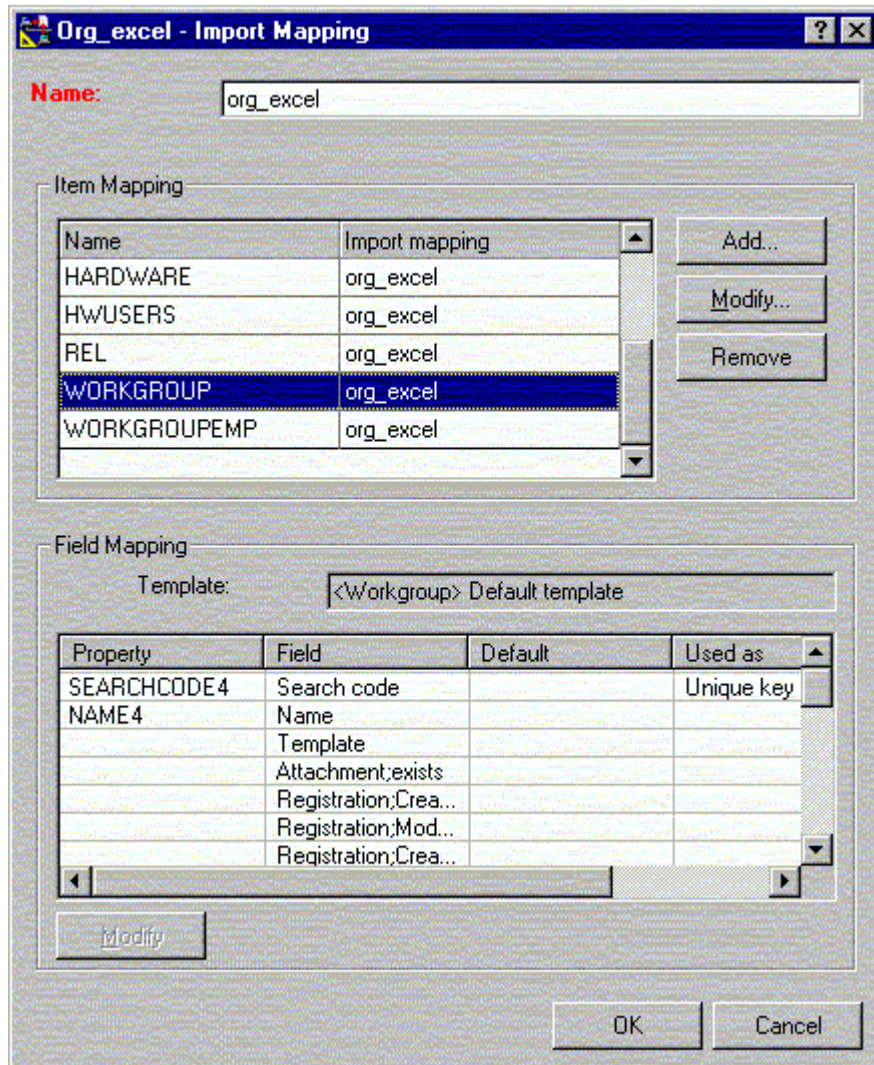
### **Import Mapping for the WORKGROUP Class**

In the import mapping the class name in brackets [WORKGROUP] is mapped to the item called Workgroup in Service Desk. The default template is used for this class. The attributes in the ATT line are mapped to fields that are available in the default template. In this case [SEARCHCODE4] is mapped to the Search code field and [NAME4] is

Examples  
Importing Data With Relations

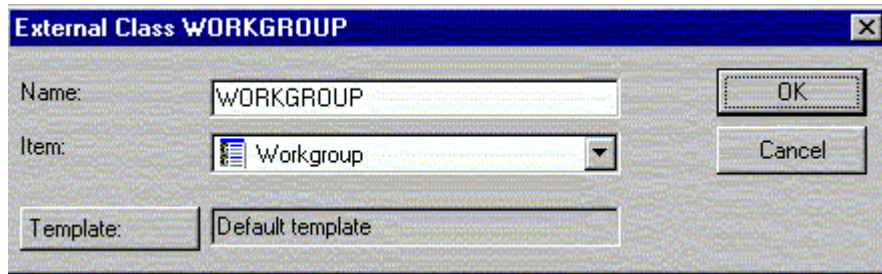
mapped to the Name field. The completed import mapping is displayed in the following dialog box:

**Figure D-32 Import Mapping - WORKGROUP**



The following dialog box shows the how the external class was mapped to the Configuration Item in Service Desk and the template selected:

**Figure D-33 Mapping External Class - WORKGROUP**



### Configuring the WORKGROUPEMP Class

This class demonstrates how to import parent-child relations. The goal of this section is to export employee search codes with their workgroup (WG) so that they can be imported into the Person item in Service Desk. The class WORKGROUPEMP is extracted from the SOURCE called EMPLOYEE. One attribute will be extracted and that is the SEARCHCODE associated with the SOURCE called EMPLOYEE. In the COLUMN field SEARCHCODE is listed and an additional COLUMN called WG is listed as the alias for the column called SEARCHCODE4.

The additional COLUMN is listed to locate the PARENT relation for this class. The PARENT line specifies the parent of this class. This is the WORKGROUPEMP class and the parent is located in the WORKGROUP table. The next line PARENT\_RELATION defines the relationship that exists between this class and the parent, the column name mentioned first, WG, is the foreign key used to identify the COLUMN of the child. The column name that follows the equal sign is the primary key of the alias of the parent, in this case it is SEARCHCODE4. The final line PARENT\_RELATION\_NAME specifies the name to be used when searching for the parent of the relation, in this case PARENT is used. The default for the PARENT\_RELATION\_NAME is PARENT.

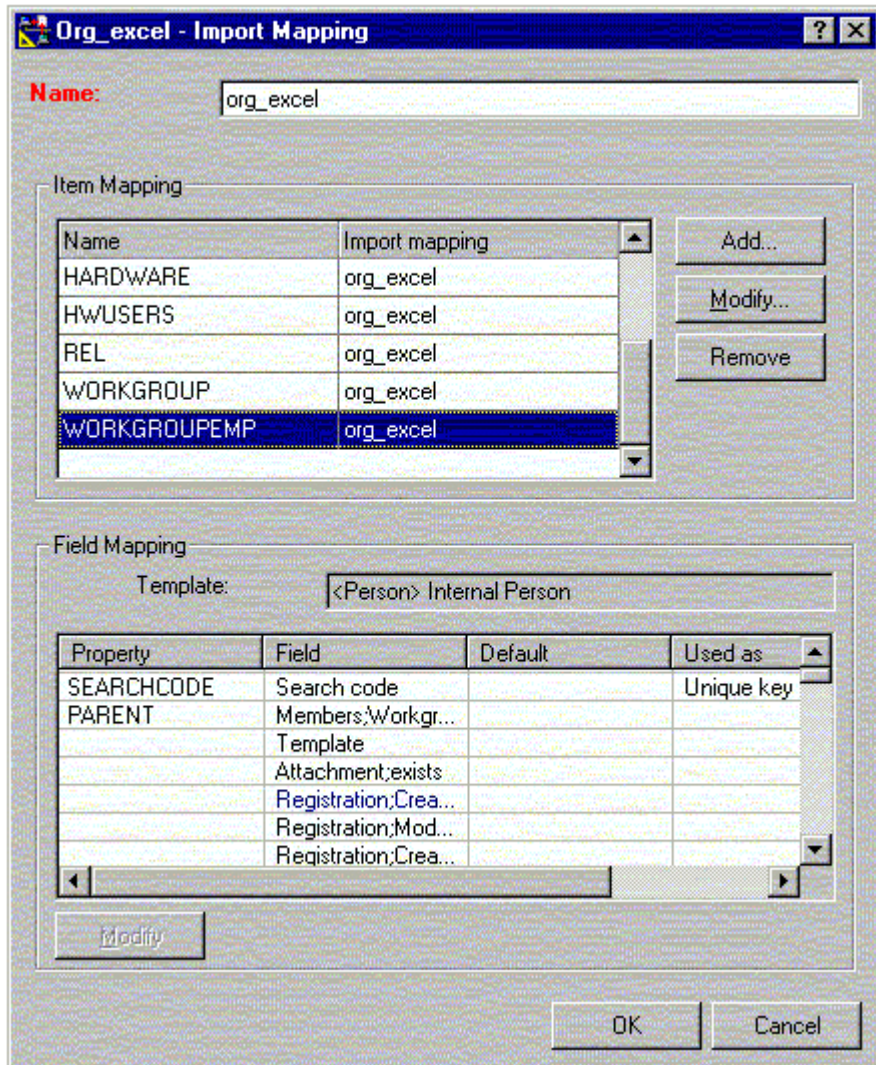
### Import Mapping for the WORKGROUPEMP Class

In the import mapping the class name in brackets [WORKGROUPEMP] is mapped to the item called Person in Service Desk, even though it came from the SOURCE called EMPLOYEE. Once the data is extracted it does not matter what the source was, what matters is what it is called in the XML file. The Internal Person template is used for this class. The attributes in the ATT line are mapped to fields that are available in

Examples  
Importing Data With Relations

the Internal Person template. In this case [SEARCHCODE] is mapped to the Search code field and the [PARENT] item is treated as an attribute itself and mapped to the parent Members;Workgroup field. The completed import mapping is displayed in the following dialog box:

**Figure D-34** Import Mapping - WORKGROUPEMP



Scrolling down in the dialog box above will show the following additional



fields:

Field	Default
Status	Active
Category	Employee

The following dialog box shows the how the external class was mapped to the `Person` item in Service Desk and the template selected:

**Figure D-35 Mapping External Class - WORKGROUPEMP**



## Importing the Data

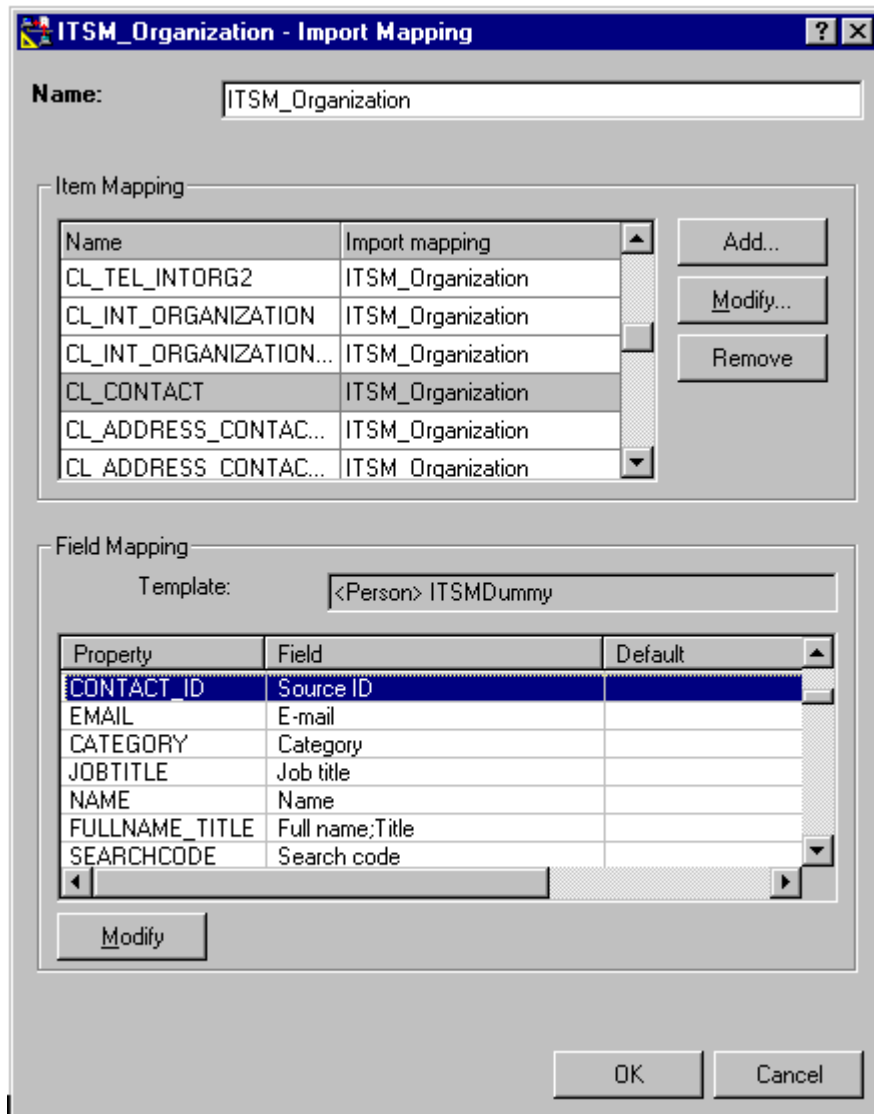
Do not import the data until after you have taken the time to verify that everything was exported correctly. You can run the task to import data from the Data Exchange user interface or from a command line, see “Using a Task to Import Data” on page 87.

## Importing Multiple Telephone Numbers

Importing more than one telephone number can be done by mapping telephone numbers to a person. If you want one telephone number to be shared by more than one person you can do this by adding a `Person` field to the `Telephone` template.

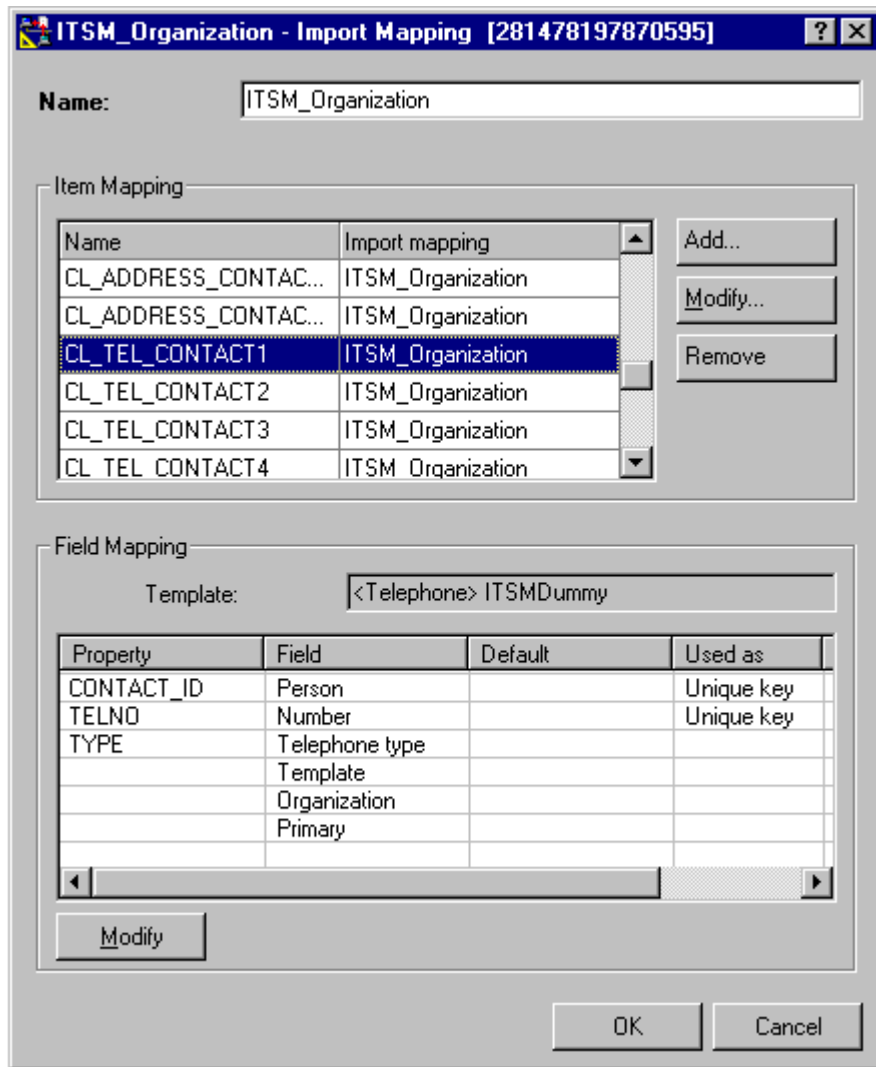
In this example multiple contact telephone numbers are mapped to contacts. The import mapping is shown in the following Import Mapping dialog box , where the class `CL_CONTACT` is mapped to the `Person` item in Service Desk. This class contains attributes that describe the contact, for example the name, and job title for each person. The attributes are mapped to Service Desk fields. The attribute `CONTACT_ID` is mapped to the `Source_ID` field in Service Desk and will be used as a search code, it is a unique key:

**Figure D-36 Import Mapping for CL\_CONTACT**



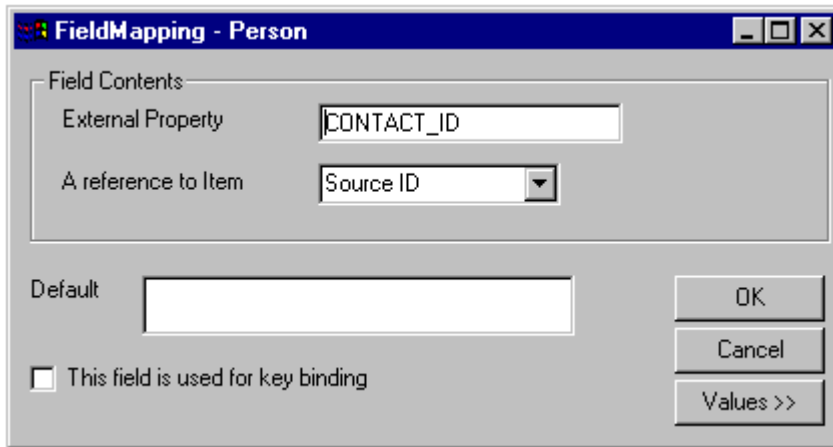
In the next dialog box, the class called CL\_TEL\_CONTACT1 is mapped to the Telephone item in Service Desk. The CL\_TEL\_CONTACT1 class contains attributes for telephone numbers:

**Figure D-37**      **Import Mapping CL\_TEL\_CONTACT1**



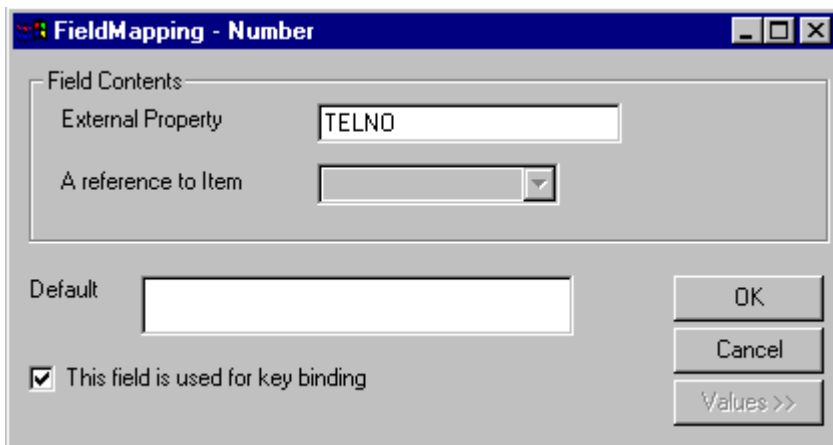
In the following dialog box, the attribute CONTACT\_ID is mapped to the Person field and uses Source ID as a search code to reference the Person item in Service Desk. This reference provides a link between the item containing the phone numbers and the item containing the people (contacts):

**Figure D-38 Field Mapping - Person Field**



The attribute `TELNO` is mapped to the Service Desk `Number` field, and set as a unique attribute with the key binding check box in the following dialog box:

**Figure D-39 Field Mapping - Number Field**



The third attribute called `TYPE` is mapped to the Service Desk field called `Telephone Type`. The field `Telephone type` contains a code table with values. Click `Values` to expand the Field Mapping dialog box for value mapping. The value `BUSINESS` for the External Property `Type` is mapped to the value `Business` in Service Desk. It is possible to enter additional

Examples  
Importing Multiple Telephone Numbers

types if you export more than one type of phone number, for example private and mobile phone numbers. The following dialog box shows the value mapping for the telephone type called BUSINESS:

**Figure D-40**      **Field Mapping - Telephone Type**

**FieldMapping - Telephone type**

Field Contents

External Property: TYPE

A reference to Item: [Dropdown]

Default: [Text Field]

This field is used for key binding

OK  
Cancel  
Values >>

Value Mapping

Value for External Attribute	Value for Internal Attribute
BUSINESS	Business

Add To List    Remove

[Text Field] Maps to [Dropdown]

## 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 MS Access, for example.

**Step 1.** Configure the `SCHEMA.INI` file for the ODBC driver.

1. Place the ASCII text file you will import in the same directory as the `SCHEMA.INI` file. The following text will be imported for this example:

```
FirstName,LastName,Address,Searchcode  
Irvine,Welsh,49 Bird road,IWELSH  
Jonathan,Cape,50 Roller Avenue,JCAPE
```

2. The `SCHEMA.INI` file is used to define the ASCII text format. Modify the `SCHEMA.INI` file for the ODBC driver as follows:

```
[sample.txt]  
ColNameHeader=True  
Format=CSVDelimited  
MaxScanRows=1  
CharacterSet=OEM  
Col1=FIRSTNAME Char Width 255  
Col2=LASTNAME Char Width 255  
Col3=ADDRESS Char Width 255  
Col4=SEARCHCODE Char Width 255
```

---

**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.

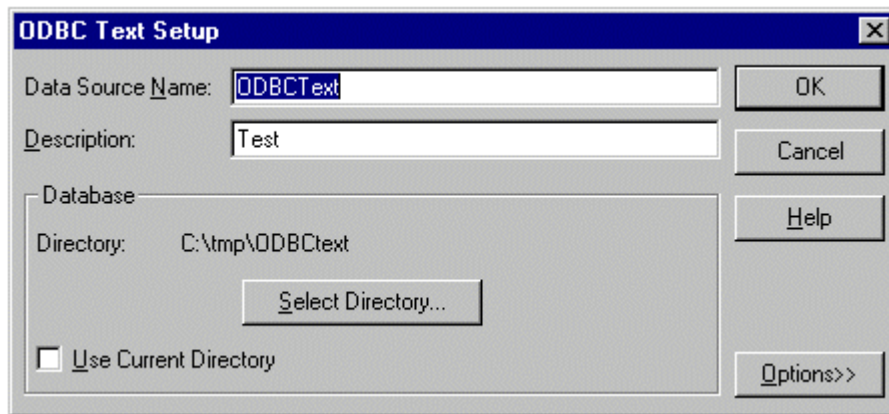
1. Start the ODBC Data Sources from the Windows Control Panel.
2. Select the `System DSN` tab and click `Add`.

## Examples

### Importing Data From an ASCII Text File

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 that the SCHEMA.ini file and the ODBCText file are located. The ODBC Text Setup dialog box should be set up similar to the following dialog box:

**Figure D-41 ODBC Text Setup**



### Step 3. Modify the configurable extractor (ini) file.

1. Copy the configurable extractor `sd_event.ini` in Service Desk and save it with a new name. For example, `ODBCtext.ini`.
2. Modify the configurable ini file as follows:

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

[ SYSTEM ]
LOG=TRUE
XML=TRUE
TXT=FALSE
DUMP=TRUE
```



```

LOG_FILE=\ODBCtext.log
OUTPUT_FILE=\ODBCtext.txt
XML_OUTPUT_FILE=\ODBCtext.xml
APPLICATION_NAME=ODBCtext
ENCODING=ISO-8859-1

[ CLASSES ]
NAME=SAMPLE

[ SAMPLE ]
SOURCE=sample.txt
ATT=[ LASTNAME ], [ FIRSTNAME ], [ ADDRESS ], [ SEARCHCODE ]
COLUMNS=[ LASTNAME ], [ FIRSTNAME ], [ ADDRESS ], [ SEARCHCODE ]
LOADTABLE=TRUE

```

**Step 4.** Export the ASCII text file data.

1. Open the Data Exchange dialog box and select the Export data from a storage device check box.
2. In the Export Mapping field enter the name of the ini file you modified. In this example it is `ODBCtext.ini`.
3. In the Exchange file field enter the name you specified for the XML file in the `ODBCtext.ini` file. In this example it is `ODBCtext.xml`.
4. Click OK to export the data into an XML file.

---

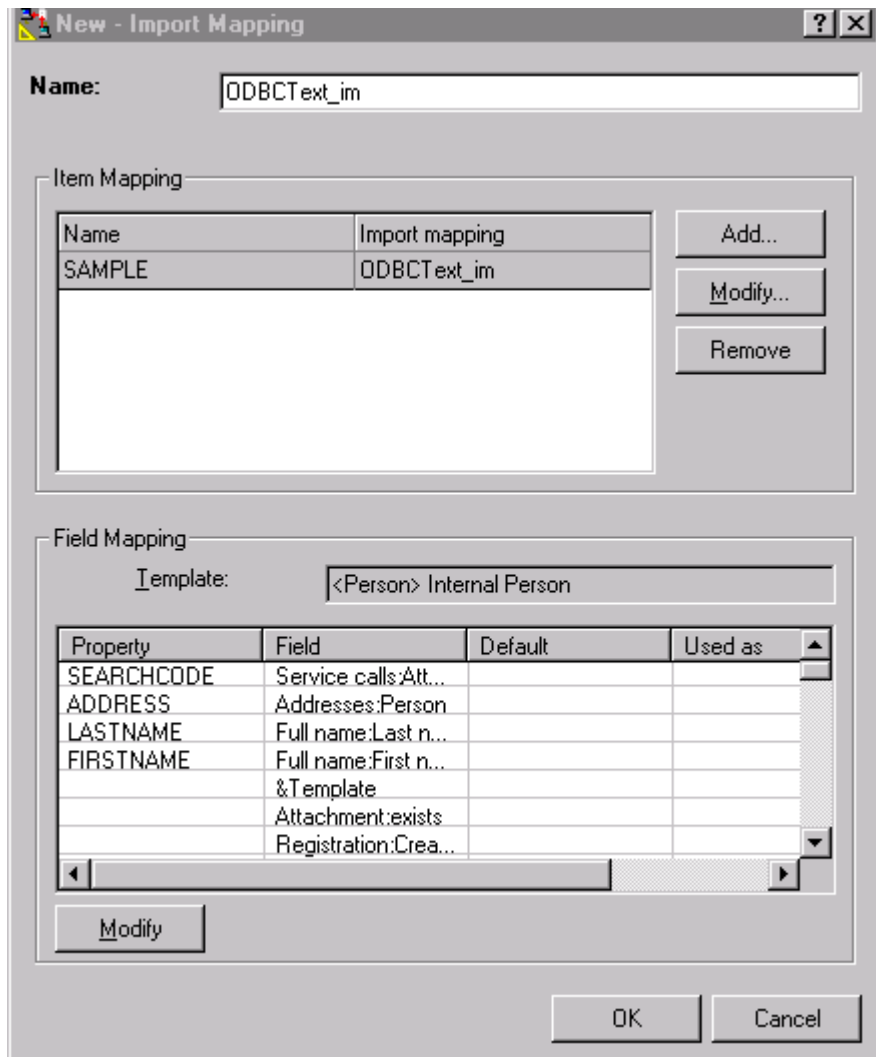
**NOTE**

You can also export the ASCII text file from a command line with:  
`sd_export -f odbctext.ini -x odbctext.xml -l odbctext.log`

---

- Step 5.** Create an import mapping for the data. The following dialog box shows the import mapping for this example:

**Figure D-42** ODBC Text file - Import Mapping



**Step 6.** Import the data.

1. Open the Data Exchange dialog box.
2. In the Exchange file field, enter the location and name of the ODBCText.xml file.

3. Select the `Import data into Service Desk database` check box.
4. Enter a Service Desk account and password created for integration purposes.
5. In the `Import mapping` field use the drop-down arrow to find the import mapping you created for importing the Excel file. In this example it is `ODBCText_IM`:
6. Select the `Debug` check box and click `OK` to import the data.
7. Check `ODBCText_IM_imp.log` to verify that the import was successful.

---

**NOTE**

You can also import the XML file from a command line with: `sd_import odbctext.xml username/password odbctext_im Y odbctext.log c\temp\data_exchange.`

---

Examples  
**Importing Data From an ASCII Text File**

---

# Glossary

## A

**action** An operation carried out as a result of the activation of a rule and the successful evaluation of a rule or conditions within the rule.

**agent** A program or process running on a remote device or computer system that responds to management requests, performs management operations, and/or sends event notification.

**API** Application programming interface. An interface that enables programmatic access to an application.

**attribute** An object characteristic or property that describes the current state of the object.

**attribute value** A value assigned to one of the properties (attributes) that are associated with an object.

## B

**batch importing** The process of importing a group (batch) of data from an external data source into the Service Desk database.

## C

**CIM-XML** Common interface model extensible markup language.

**condition** A rule element consisting of a condition type and a set of parameter values to which a source value is compared to determine if the condition holds true or not, (eg. "IP Address is 15.2.112\*").

**configuration item** An item belonging to the technical infrastructure of an organization. A configuration item may consist of other configuration items, and may be part of other configuration items. For example, a PC, an application program, a network, a work space with desks and chairs. Configuration items are stored in the application database.

**class** *see object*

## D

**database rule** *see rule*

**daemon** A software process that runs continuously in the background and provides services upon request.

**data exchange file** A term used to refer to the files that are exported during the data exchange process. The data exchange process creates three data exchange files: a text file an XML file and a log file.

**DTD** Document type definition. A DTD is a set of syntax rules for mark-up tags and their

---

interpretation within an XML or HTML file. A DTD provides specific information about the tags used in the document; the order those tags should appear, which tags can appear inside other ones, and which tags have attributes, for example. A DTD defines the relationship between document elements.

## E

**event** An event is an unsolicited notification such as an SNMP trap, node down notification or TL1 event, generated by an agent or process in a managed object or by a user action. Events usually indicate a change in the state of a managed object or cause an action to occur.

**event communicator** Includes the `sd_event` program and the Rule Manager agent, which are necessary for sending and receiving events.

**event queue** The process of putting multiple events in order so that they can be executed one by one.

**event storm** When hundreds of events occur at the same time it is referred to as an event storm. Queuing tools are used to organize the events so that they do not flood the system all at one time.

**extractor** A program used for

exporting 'extracting' information from a data source. The extractor is configurable through the use of an \*.ini file.

## H

**HTML** Hypertext markup language. A standard generalized markup language (SGML) document type definition (DTD) that provides a collection of platform-independent styles (indicated by markup tags) to define the various components of a World Wide Web (WWW) document.

**HTTP** Hypertext transfer protocol. The protocol that World Wide Web clients and servers use to communicate.

## I

**integration account** A Service Desk account created specifically for the purposes of integration. Integration accounts normally are not given access to the user interface.

**item** *See object*

## K

**key binding** The process of assigning at least one attribute the role of identification for an item. A typical key binding attribute is a search code or a serial number.

---

**key value** *See key binding*

## M

**mapping** The process of matching external classes, properties and values with Service Desk items, attributes and values. Mapping is done so that exported information can be formatted in such a way that it can be correctly imported into Service Desk.

**message** A structured readable notification that is generated as a result of an event, the evaluation of one or more events relative to specified conditions, or a change in application, system, network, or service events.

**message annotation** Message annotations can be written for each message, and often describe the actions that were taken to solve the problem. An annotation can be created and reviewed by the operator or administrator for any message.

**modus** The mode being used to make a change in a database. For example, inserting a record, deleting a record or updating a record.

## O

**object** A managed logical or physical resource, or a group of such physical resources that exist in a managed environment.

Example of objects are a network, a computer, an interface, and a printer.

**ODBC** Open database connectivity. Software that enables a program to communicate with a particular RDBMS. ODBC permits database independence by providing an interface to a set of drivers for the RDBMS' supported by OpenView.

## outbound service events

Events that are sent from the Service Desk application.

## P

**property** A characteristic or attribute of a class, object, or item.

## Q

**queue** A waiting line in which unsatisfied requests (events) are placed until a resource becomes available to execute them.

## R

**rule** A rule is the combination of one or more actions and the set of conditions that determine when the action will take place.

**rule manager** Maintains rules and is responsible for pulling event information from the event queue to determine if a rule exists that applies to the event.

---

**rule manager agent** *see agent*

## **S**

**service event** *see event*

**SNMP** Simple network management protocol. A protocol running above TCP/IP used to communicate network management information.

**SNMP trap** An unconfirmed event, generated by an SNMP agent in response to some internal state change or fault condition, which conforms to the protocol specified.

## **X**

**XML** Extensible markup language. Similiar to HTML except that it provides more semantic information. The file format is used to represent data, and for describing the data structure. The tags used make it possible to indicate what kind of data each tag contains, rather than indicating only how something should look.



---

# Index

---

## A

Acceptance Filter  
  ManageX, 170  
accounts, integration, 95  
attributes  
  import mapping, 62  
  ITO import mapping, 158  
auditing, 123

## C

CIM-XML, viewing, 57  
classes, import mapping, 62  
COM objects  
  ManageX, 175  
command syntax, 92  
  service events, 101  
configuration  
  extractor, 37  
  importing, 31, 62  
  keywords, 39  
  service events, 103

## D

data exchange  
  concept, 27  
  exporting data procedures, 51  
  files, 53  
  importing data, 87  
  scheduling, 91  
  tasks, 90  
database rules  
  ManageX, 168  
database structure, checking, 35  
debug mode, 124  
document conventions, 21

## E

error log, 105  
event storms, 107  
example service events. See  
  service events

## examples

  configuration files, 60  
  exported XML file, 54  
  ITO integration, 146  
  queuing, 160, 176  
export files, 53  
exporting  
  command syntax, 92  
  data, 37  
  new task, 53  
  procedures, 51  
  process, 49  
  what to export, 35  
extractor, 37

## I

import mapping  
  classes, 62  
  relations, 63  
  templates, 62  
  values, 81  
import mapping templates, 60  
importing  
  command syntax, 92  
  procedures, 87  
importing mapping  
  attributes, 62  
installation  
  ITO tar file, 151  
integration accounts, 95  
ITO  
  attribute mapping, 158  
  example integration, 146  
  installation, 151  
  queuing example, 160, 176  
  tar file, 151

## K

key binding, 62  
key values, 62  
keywords, 39

## L

log files, 124

## M

ManageX  
  Acceptance Filter, 170  
  COM objects, 175  
  database rules, 168  
  multiple servers, 167  
  string substitutes, 174  
multiple servers  
  ManageX, 167

## N

Network Node Manager  
  command line, 135  
  ODBC, 47  
  variables, 137

## O

ODBC (open database  
  connectivity), 46  
  establishing link, 46  
  Network Node Manager, 47

## Q

queuing events, 107

## R

related publications, 18  
relations, import mapping, 63  
required fields, 63  
resending service events, 105

## S

scheduling tasks, 91  
sd\_event.exe, 101  
sending service events, 111  
service events  
  command line, 101

configuration file, 103  
NNM example, 134  
outbound, 111  
queuing, 107  
resending, 105  
switches, 101  
SQL statements, 49  
string substitutes  
  ManageX, 174  
switches, service events, 101

## T

tasks  
  creating, 90  
  exporting, 53  
  scheduling, 91  
templates, import mapping, 62  
troubleshooting  
  debug mode, 124  
  log files, 124  
  program files, 125  
  viewing XML files, 57  
typographic conventions, 21

## V

value mapping, 81  
variables  
  Network Node Manager, 137  
viewing, XML files, 57

## X

XML (extensible markup  
  language)  
  about, 54  
  CIM-XML, 53  
  example XML file, 54  
  export files, 53  
  viewing export files, 57