This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http://www.opensource.org/licenses/bsd-license.php) the MIT License (http://www.opensource.org/licenses/mit-license.php) and the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0).

This product includes software copyright © 2003-2006 Joe WaInes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://xstream.codehaus.org/license.html. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/.

This Software is protected by U.S. Patent Numbers 5,794,246; 6,014,670; 6,016,501; 6,029,178; 6,032,158; 6,035,307; 6,044,374; 6,092,086; 6,208,990; 6,339,775; 6,640,226; 6,789,096; 6,820,077; 6,823,373; 6,850,947; 6,895,471; 7,117,215; 7,162,643; 7,243,110; 7,254,590; 7,281,001; 7,496,568; 7,523,121; 7,584,422; 7,676,516; 7,720,842; 7,774,791, international Patents and other Patents Pending.

DISCLAIMER: Informatica Corporation provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose and non-infringement. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

Part Number: PCX-DUG-95100-0001
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>x</td>
</tr>
<tr>
<td>Informatica Resources.</td>
<td>x</td>
</tr>
<tr>
<td>Informatica MySupport Portal.</td>
<td>x</td>
</tr>
<tr>
<td>Informatica Documentation.</td>
<td>x</td>
</tr>
<tr>
<td>Informatica Web Site.</td>
<td>x</td>
</tr>
<tr>
<td>Informatica Marketplace.</td>
<td>x</td>
</tr>
<tr>
<td>Chapter 1: Introduction to PowerCenter Express</td>
<td>1</td>
</tr>
<tr>
<td>PowerCenter Express Overview.</td>
<td>1</td>
</tr>
<tr>
<td>PowerCenter Express Example</td>
<td>1</td>
</tr>
<tr>
<td>PowerCenter Express Architecture.</td>
<td>2</td>
</tr>
<tr>
<td>Data Integration Process.</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 2: Informatica Developer.</td>
<td>5</td>
</tr>
<tr>
<td>Informatica Developer Overview.</td>
<td>5</td>
</tr>
<tr>
<td>Start Informatica Developer.</td>
<td>5</td>
</tr>
<tr>
<td>Starting Developer Tool on a Local Machine.</td>
<td>6</td>
</tr>
<tr>
<td>Starting Developer Tool on a Remote Machine.</td>
<td>6</td>
</tr>
<tr>
<td>Informatica Developer User Interface.</td>
<td>7</td>
</tr>
<tr>
<td>Informatica Developer Welcome Page.</td>
<td>8</td>
</tr>
<tr>
<td>Cheat Sheets.</td>
<td>8</td>
</tr>
<tr>
<td>Informatica Preferences.</td>
<td>9</td>
</tr>
<tr>
<td>Informatica Marketplace.</td>
<td>9</td>
</tr>
<tr>
<td>Setting Up Informatica Developer.</td>
<td>9</td>
</tr>
<tr>
<td>Adding the Domain and Repository.</td>
<td>10</td>
</tr>
<tr>
<td>Selecting the Default Data Integration Service.</td>
<td>13</td>
</tr>
<tr>
<td>The Model Repository.</td>
<td>14</td>
</tr>
<tr>
<td>Objects in Informatica Developer.</td>
<td>14</td>
</tr>
<tr>
<td>Object Properties.</td>
<td>15</td>
</tr>
<tr>
<td>Connecting to a Model Repository.</td>
<td>16</td>
</tr>
<tr>
<td>Projects</td>
<td>16</td>
</tr>
<tr>
<td>Creating a Project.</td>
<td>16</td>
</tr>
<tr>
<td>Filter Projects.</td>
<td>17</td>
</tr>
<tr>
<td>Project Permissions.</td>
<td>17</td>
</tr>
<tr>
<td>Permissions for External Objects.</td>
<td>18</td>
</tr>
<tr>
<td>Permissions for Dependent Object Instances.</td>
<td>18</td>
</tr>
<tr>
<td>Parent Object Access.</td>
<td>19</td>
</tr>
<tr>
<td>Assigning Permissions.</td>
<td>19</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Folders</td>
<td>20</td>
</tr>
<tr>
<td>Creating a Folder</td>
<td>20</td>
</tr>
<tr>
<td>Search</td>
<td>20</td>
</tr>
<tr>
<td>Searching for Objects and Properties</td>
<td>21</td>
</tr>
<tr>
<td>Workspace Editor</td>
<td>21</td>
</tr>
<tr>
<td>Find in Editor</td>
<td>22</td>
</tr>
<tr>
<td>Validation Preferences</td>
<td>23</td>
</tr>
<tr>
<td>Grouping Error Messages</td>
<td>23</td>
</tr>
<tr>
<td>Limiting Error Messages</td>
<td>23</td>
</tr>
<tr>
<td>Copy</td>
<td>24</td>
</tr>
<tr>
<td>Copying an Object</td>
<td>24</td>
</tr>
<tr>
<td>Saving a Copy of an Object</td>
<td>24</td>
</tr>
<tr>
<td>Tags</td>
<td>25</td>
</tr>
<tr>
<td>Creating a Tag</td>
<td>25</td>
</tr>
<tr>
<td>Assigning a Tag</td>
<td>25</td>
</tr>
<tr>
<td>Viewing Tags</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 3: Connections</td>
<td>27</td>
</tr>
<tr>
<td>Connections Overview</td>
<td>27</td>
</tr>
<tr>
<td>Connection Explorer View</td>
<td>28</td>
</tr>
<tr>
<td>IBM DB2 Connection Properties</td>
<td>29</td>
</tr>
<tr>
<td>JDBC Connection Properties</td>
<td>30</td>
</tr>
<tr>
<td>Microsoft SQL Server Connection Properties</td>
<td>31</td>
</tr>
<tr>
<td>ODBC Connection Properties</td>
<td>32</td>
</tr>
<tr>
<td>Oracle Connection Properties</td>
<td>33</td>
</tr>
<tr>
<td>Web Services Connection Properties</td>
<td>34</td>
</tr>
<tr>
<td>Connection Management</td>
<td>35</td>
</tr>
<tr>
<td>Creating Connections</td>
<td>36</td>
</tr>
<tr>
<td>Showing Connections</td>
<td>36</td>
</tr>
<tr>
<td>Refreshing the Connections List</td>
<td>37</td>
</tr>
<tr>
<td>Editing a Connection</td>
<td>37</td>
</tr>
<tr>
<td>Copying a Connection</td>
<td>37</td>
</tr>
<tr>
<td>Deleting a Connection</td>
<td>38</td>
</tr>
<tr>
<td>Chapter 4: Physical Data Objects</td>
<td>39</td>
</tr>
<tr>
<td>Physical Data Objects Overview</td>
<td>39</td>
</tr>
<tr>
<td>Relational Data Objects</td>
<td>40</td>
</tr>
<tr>
<td>Key Relationships</td>
<td>41</td>
</tr>
<tr>
<td>Creating a Read Transformation from Relational Data Objects</td>
<td>42</td>
</tr>
<tr>
<td>Importing a Relational Data Object</td>
<td>43</td>
</tr>
<tr>
<td>Customized Data Objects</td>
<td>43</td>
</tr>
<tr>
<td>Key Relationships</td>
<td>45</td>
</tr>
<tr>
<td>Customized Data Object Write Properties</td>
<td>45</td>
</tr>
</tbody>
</table>
Chapter 5: Schema Object .......................................................... 78

Schema Object Overview .......................................................... 78

Schema Object Overview View .................................................... 78

Schema Files ........................................................................... 79

Schema Object Schema View ...................................................... 79

Namespace Properties .............................................................. 80

Element Properties ................................................................... 81

Simple Type Properties ............................................................ 82
Chapter 6: Profiles ............................................................... 90
Profiles Overview ............................................................ 90
Profile Views ............................................................. 90
Column Profiling Process .................................................. 91
Column Profile Options .................................................... 92
Rules .............................................................. 93
Filtering Options ....................................................... 93
Sampling Properties .................................................... 94
Creating a Profile ......................................................... 94
Column Profile Results ..................................................... 94
Column Value Properties ................................................ 95
Column Pattern Properties ............................................... 95
Column Statistics Properties ............................................... 96
Exporting Profile Results from Informatica Developer ................. 96
Mapplet and Mapping Profiles ................................................. 97
Generating a Mapping from a Profile ........................................ 97
Running a Profile on a Mapplet or Mapping Object ......................... 97
Comparing Profiles for Mapping or Mapplet Objects ..................... 98

Chapter 7: Logical View of Data .................................................... 99
Logical View of Data Overview ................................................ 99
Logical Data Object Model Example ...................................... 100
Developing a Logical View of Data ............................................ 100
Logical Data Object Models ................................................. 101
Creating a Logical Data Object Model ...................................... 101
Importing a Logical Data Object Model from a Modeling Tool ........... 102
Logical Data Object Model Properties ....................................... 102
CA ERwin Data Modeler Import Properties ................................ 102
IBM Cognos Business Intelligence Reporting - Framework Manager Import Properties ........ 103
SAP BusinessObjects Designer Import Properties ....................... 104
Sybase PowerDesigner CDM Import Properties .......................... 105
Sybase PowerDesigner OOM 9.x to 15.x Import Properties ............... 106
# Table of Contents

- Dependency Types ................................................................. 128
- Link Path Dependencies ......................................................... 128
- Implicit Dependencies ............................................................. 128
- Propagated Port Attributes by Transformation .......................... 128
- Mapping Validation ............................................................... 129
- Connection Validation ............................................................ 130
- Expression Validation ............................................................ 130
- Object Validation ................................................................. 130
- Validating a Mapping ............................................................. 130
- Running a Mapping .................................................................. 130
- Segments .................................................................................. 131
- Copying a Segment .................................................................. 131

**Chapter 10: Mapplets.** .......................................................... 132

- Mapplets Overview ............................................................... 132
- Mapplet Types .......................................................................... 133
- Mapplets and Rules ............................................................... 134
- Mapplet Input and Output ....................................................... 134
  - Mapplet Input ........................................................................ 134
  - Mapplet Output ..................................................................... 135
- Creating a Mapplet ............................................................... 135
- Validating a Mapplet ............................................................. 135

**Chapter 11: Viewing Data.** ..................................................... 136

- Viewing Data Overview .......................................................... 136
- Configurations ......................................................................... 136
  - Configuration Properties ...................................................... 137
  - Data Viewer Configurations .................................................. 139
  - Mapping Configurations ...................................................... 141
  - Updating the Default Configuration Properties ....................... 143
  - Troubleshooting Configurations ............................................ 144
- Exporting Data ......................................................................... 145
- Logs ........................................................................................ 145
  - Log File Format ..................................................................... 146
- Monitoring Jobs from the Developer Tool ................................... 146

**Chapter 12: Workflows.** ......................................................... 147

- Workflows Overview ............................................................. 147
- Creating a Workflow .............................................................. 148
- Workflow Objects ..................................................................... 148
  - Events .................................................................................. 148
  - Tasks ................................................................................... 149
  - Exclusive Gateways ............................................................. 149
Parameter File Structure ................................................  168
Project Element. ......................................................  169
Application Element. ...................................................  170
Rules and Guidelines for Parameter Files. ......................................  170
Sample Parameter File. .................................................  170
Creating a Parameter File. ................................................  172
Running a Mapping with a Parameter File. ................................................  173

Chapter 15: Object Import and Export ............................................ 174
Object Import and Export Overview. ........................................... 174
Import and Export Objects. .................................................. 174
Object Export. ........................................................... 175
Exporting Objects. ..................................................... 176
Object Import. ........................................................... 176
Importing Projects. ..................................................... 176
Importing Objects. ..................................................... 177

Chapter 16: Performance Tuning ................................................  180
Optimizer Levels. ......................................................... 180
Optimization Methods Overview. ............................................... 181
  Early Projection Optimization Method. ......................................... 181
  Early Selection Optimization Method. ......................................... 181
  Predicate Optimization Method. ............................................ 182
  Cost-Based Optimization Method. ........................................... 182
  Semi-Join Optimization Method. .......................................... 183
Full Optimization and Memory Allocation. ........................................ 183
Setting the Optimizer Level for a Developer Tool Mapping. ...................... 184
Setting the Optimizer Level for a Deployed Mapping. .................................. 184

Chapter 17: Pushdown Optimization ............................................. 185
Pushdown Optimization Overview. ........................................... 185
Pushdown Optimization to Sources. ........................................... 186
  Pushdown Optimization to Native Sources. ....................................... 186
  Pushdown Optimization to ODBC Sources. ....................................... 186
Pushdown Optimization Expressions. ......................................... 187
  Functions. .......................................................... 187
  Operators. .......................................................... 191
Comparing the Output of the Data Integration Service and Sources. .......... 191

Appendix A: Datatype Reference ................................................ 193
Datatype Reference Overview. ............................................... 193
Transformation Datatypes. ................................................ 194
  Integer Datatypes. ..................................................... 194
Binary Datatype ...................................................... 196
Date/Time Datatype ................................................... 196
Decimal and Double Datatypes ............................................ 197
String Datatypes ...................................................... 198
Flat File and Transformation Datatypes ....................................... 198
IBM DB2 and Transformation Datatypes ......................................... 199
    Unsupported IBM DB2 Datatypes ........................................... 199
JDBC and Transformation Datatypes ........................................... 200
Microsoft SQL Server and Transformation Datatypes ............................ 201
    Unsupported Microsoft SQL Server Datatypes .................................. 202
ODBC and Transformation Datatypes ........................................... 203
Oracle and Transformation Datatypes ........................................... 204
    Number(P,S) Datatype .................................................. 205
    Char, Varchar, Clob Datatypes ............................................ 205
    Unsupported Oracle Datatypes ............................................ 205
XML and Transformation Datatypes ............................................ 206
Converting Data ......................................................... 207
    Port-to-Port Data Conversion ............................................. 208

Index ............................................................... 209
Preface

The Informatica PowerCenter Express User Guide is written for data integration developers. This guide assumes that you have an understanding of flat file and relational database concepts, the database engines in your environment, and data integration concepts.

Informatica Resources

Informatica MySupport Portal

As an Informatica customer, you can access the Informatica MySupport Portal at http://mysupport.informatica.com. The site contains product information, user group information, newsletters, access to the Informatica How-To Library, the Informatica Knowledge Base, the Informatica Multimedia Knowledge Base, Informatica Product Documentation, and access to the Informatica user community.

Informatica Documentation

The Informatica Documentation team takes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at info_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

The Documentation team updates documentation as needed. To get the latest documentation for your product, navigate to Product Documentation from http://mysupport.informatica.com.

Informatica Web Site

You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica Marketplace

The Informatica Marketplace is a forum where developers and partners can share solutions that augment, extend, or enhance data integration implementations. By leveraging any of the hundreds of solutions available on the Marketplace, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at http://www.informaticamarketplace.com.
CHAPTER 1

Introduction to PowerCenter Express

This chapter includes the following topics:
- PowerCenter Express Overview, 1
- PowerCenter Express Architecture, 2
- Data Integration Process, 3

PowerCenter Express Overview

Use PowerCenter Express to design and implement data integration solutions.

You can use PowerCenter Express to extract data from multiple sources, transform the data according to business logic that you build in the client application, and load the transformed data to targets. You can also run a profile to analyze the structure and content of your data, and to determine the quality of your data.

You can access data in relational databases, flat files, web services, and social media web sites.

PowerCenter Express includes the Informatica domain, application services that process data, repositories to store metadata, Informatica Administrator (the Administrator tool), and Informatica Developer (the Developer tool).

The Administrator tool, the repositories, and application services run on a single machine. The Developer tool can run on one or more machines.

PowerCenter Express Example

Organizations can use PowerCenter Express to perform data integration tasks. For example, use PowerCenter Express to consolidate data.

An organization wants to consolidate data from three flat files that contain customer records from different geographic areas. After analyzing the content of the data, the organization wants to load the data to a relational database table.

To complete this task, the organization uses the Developer tool to run a profile on the data, consolidate the data, and write the consolidated data to a relational database. Then, the organization uses the Administrator tool to monitor the progress of the profile job and the progress of the workflow that consolidates the data.
PowerCenter Express Architecture

The PowerCenter Express tools, application services, and repositories are components that run within the Informatica domain.

The Informatica domain is the fundamental administrative unit in Informatica. The Informatica domain contains the following components:

- Application clients. A group of clients that you use to access underlying Informatica functionality. Application clients make requests to the Service Manager or application services.
- Application services. A group of services that represent server-based functionality. You configure the application services that are required by the application clients that you use.
- Repositories. A group of relational databases that store metadata about objects and processes required to handle user requests from application clients.
- Service Manager. A service that is built in to the domain to manage all domain operations. The Service Manager runs the application services and performs domain functions including authentication, authorization, and logging.

The following figure shows the PowerCenter Express components that run within the Informatica domain:

The Informatica domain includes the following PowerCenter Express components:

**Informatica Administrator**

Informatica Administrator (the Administrator tool) is an application client that consolidates the administrative tasks for domain objects such as services, connections, and licenses. You manage the domain and the security of the domain through the Administrator tool.

**Informatica Developer**

Informatica Developer (the Developer tool) is an application client that you can use to design and implement data integration solutions.
Data Integration Service

The Data Integration Service is an application service that performs data integration tasks for Informatica Developer and external clients. Data integration tasks include previewing data and running profiles, mappings, and workflows.

Model Repository Service

The Model Repository Service is an application service that manages the Model repository.

Model repository

The Model repository is a relational database that stores the metadata for projects. The Model repository also stores run-time and configuration information for applications that are deployed to a Data Integration Service.

Domain configuration repository

The domain configuration repository is a set of domain metadata tables stored in a relational database. Each time you make a change to the domain, the Service Manager writes the change to the domain configuration repository.

Profiling warehouse

The profiling warehouse is a relational database that the Data Integration Services uses to store profiling results.

Data Integration Process

Use the PowerCenter Express client applications to complete your data integration solutions. Use the Developer tool to create connections, import metadata, run profiles, create mappings, and run mappings as part of workflows. Then, use the Administrator tool to monitor workflow progress.

Step 1. Create connections

Create connections to access data from relational databases, third-party web services, or social media web sites. Create a connection to create data objects, preview data, run profiles, and run mappings. The Developer tool uses the connection when you import a data object. The Data Integration Service uses the connection when you preview data, run profiles, and run mappings.

Step 2. Import metadata to create data objects

Import metadata to create data objects for sources and targets that you want to use in a mapping. When you develop a mapping, you can use data objects to define the input and output of the mapping.

Step 3. Run a profile

Run a profile to analyze the structure and content of your data, and to determine the quality of your data. When you run a profile, the Data Integration Service applies the profiling rules and runs the profile.

Step 4. Develop mappings

Develop mappings to implement data integration tasks. A mapping is a set of inputs and outputs that represent the data flow between sources and targets. Link the sources and targets with transformation objects that define the rules for data transformation. The Data Integration Service uses the instructions configured in the mapping to read, transform, and write data. You can add a mapping as a task in a workflow.
Step 5. Create and run workflows

Create a workflow to define a sequence of events, tasks, and decisions based on a business process. Then, deploy the workflow to the Data Integration Service and run the workflow. The Data Integration Service uses the instructions configured in the workflow to run the objects.

Step 6. Monitor workflows

Monitor the workflow instance run on the Monitoring tab of the Administrator tool. The Monitoring tab shows the status of running workflow and workflow object instances. You can abort or cancel a running workflow instance in the Monitoring tool. You can also use the Monitoring tool to view logs for workflow instances and to view workflow reports.
This chapter includes the following topics:

- Informatica Developer Overview, 5
- Start Informatica Developer, 5
- Informatica Developer User Interface, 7
- Setting Up Informatica Developer, 9
- The Model Repository, 14
- Projects, 16
- Project Permissions, 17
- Folders, 20
- Search, 20
- Workspace Editor, 21
- Validation Preferences, 23
- Copy, 24
- Tags, 25

Informatica Developer Overview

The Developer tool is an application that you use to design and implement data integration solutions.

You can use the Developer tool to import metadata, create connections, and create logical data objects. You can also use the Developer tool to create and run profiles, mappings, workflows.

Start Informatica Developer

If the Developer tool is installed on a local machine, use the Windows Start menu to start the tool. If the Developer tool is installed on a remote machine, use the command line to start the tool.
Starting Developer Tool on a Local Machine

Use the Windows Start menu to start the Developer tool installed on a local machine.

1. From the Windows Start menu, click All Programs > Informatica PowerCenter Express > Launch Informatica Developer.
   The first time you run the Developer tool, the Welcome page displays multiple icons. The Welcome page does not appear when you run the Developer tool again.

2. Click Workbench.
   If you installed Informatica services and Informatica client separately, the first time you start the Developer tool, you must set up the tool by adding a domain, adding a Model repository, and selecting a default Data Integration Service.
   If you installed Informatica services and Informatica client at the same time on the same machine, the Object Explorer view displays the Model repository with the default name ModelRepository.

Starting Developer Tool on a Remote Machine

Use the command line to start the Developer tool installed on a remote machine.

When the Developer tool is installed on a remote machine, you might not have write access to the installation directory. You must specify a workspace directory on your local machine where the Developer tool can write temporary files. An administrator can configure the default local workspace directory for all users. You can override the default directory when you start the Developer tool.

If the configured local workspace directory does not exist, the Developer tool creates the directory when it writes temporary files.

1. Open a command prompt.
2. Enter the command to start the Developer tool. You can use the default local workspace directory or override the default directory.
   ✷ To use the default local workspace directory, enter the following command:
     `\\<remote installation directory>\developer.exe`
     For example:
     `\\MyRemoteMachine\Informatica\PCEXpress\client\DeveloperClient\developer.exe`
   ✷ To override the default local workspace directory, enter the following command:
     `\\<remote installation directory>\developer.exe -data <local workspace directory>`
     For example:
     `\\MyRemoteMachine\Informatica\PCEXpress\client\DeveloperClient\developer.exe -data C:\temp \MyWorkspace`
   Folder names in the local workspace directory cannot contain the number sign (#) character. If folder names in the local workspace directory contain spaces, enclose the full directory in double quotes.

The first time you run the Developer tool, the Welcome page displays multiple icons. The Welcome page does not appear when you run the Developer tool again.

3. Click Workbench.
   If you installed Informatica services and Informatica client separately, the first time you start the Developer tool, you must set up the tool by adding a domain, adding a Model repository, and selecting a default Data Integration Service.
   If you installed Informatica services and Informatica client at the same time on the same machine, the Object Explorer view displays the Model repository with the default name ModelRepository.
The Developer tool is an application that you use to design and implement data integration solutions. The Developer tool workbench includes an editor and views.

You edit objects, such as mappings, in the editor. The Developer tool displays views, such as the **Properties** view, based on which object is in focus in the editor and your selection of which views you want to display.

The following figure shows the Developer tool workbench:

1. Outline view
2. Object Explorer view
3. Editor
4. Connection Explorer view
5. Properties view

The Developer tool displays the following views by default:

**Outline view**
Displays objects that are dependent on an object selected in the **Object Explorer** view. By default, this view appears in the bottom left area of the Developer tool.

**Object Explorer view**
Displays projects, folders, and the objects within the projects and folders. By default, this view appears in the top left area of the Developer tool.

**Connection Explorer view**
Displays connections to relational databases. By default, this view appears in the top right area of the Developer tool.
Properties view
Displays the properties for an object that is in focus in the editor. By default, this view appears in the bottom area of the Developer tool.

You can hide views and move views to another location in the Developer tool workbench. Click Window > Show View to select the views you want to display.

The Developer tool workbench also displays the following views:
Cheat Sheets view
Displays the cheat sheet that you open. To open a cheat sheet, click Help > Cheat Sheets and select a cheat sheet.

Data Viewer view
Displays source data and previews the output of a transformation.

Help view
Displays context-sensitive online help.

Progress view
Displays the progress of operations in the Developer tool, such as a mapping run.

Search view
Displays the search results. You can also launch the search options dialog box.

Tags view
Displays tags that define an object in the Model repository based on business usage.

Validation Log view
Displays object validation errors.

Informatica Developer Welcome Page
The first time you open the Developer tool, the Welcome page appears. Use the Welcome page to learn more about the Developer tool, set up the Developer tool, and to start working in the Developer tool.

The Welcome page displays the following options:
Overview
Click the Overview button to get an overview of PowerCenter Express and a cheat sheet that includes the first steps to begin using PowerCenter Express.

Tutorials
Click the Tutorials button to see cheat sheets for data integration tasks.

Web Resources
Click the Web Resources button for links to Informatica resources that you can access on the web. Web resources include product documentation, how-to articles, and video tutorials.

Click Help > Welcome to access the welcome page after you close it.

Cheat Sheets
The Developer tool includes cheat sheets as part of the online help. A cheat sheet is a step-by-step guide that helps you complete one or more tasks in the Developer tool.
When you follow a cheat sheet, you complete the tasks and see the results. For example, you can complete a cheat sheet to import and preview a physical data object.

To access cheat sheets, click Help > Cheat Sheets.

Informatica Preferences

The Preferences dialog box contains settings for the Developer tool and for the Eclipse platform.

Use the Informatica preferences to manage settings in the Developer tool. For example, use Informatica preferences to manage configurations, connections, transformation settings, tags, or available Data Integration Services.

The Developer tool is built on the Eclipse platform. The Preferences dialog box also includes preferences to manage settings for the Eclipse platform. Informatica supports only the Informatica preferences.

To access Informatica preferences, click Window > Preferences. In the Preferences dialog box, select Informatica.

Informatica Marketplace

The Informatica Marketplace provides prebuilt solutions to augment, extend, or enhance your data integration implementation.

To access Informatica Marketplace, click Marketplace on the toolbar. The Marketplace view appears in the Developer tool.

You must register as a user before you can log in to the Marketplace for the first time.

After you log in, you can view links to prebuilt solutions in the editor. You can search for a solution in the Marketplace search box and view the search results to find the solution. A solution might contain mappings, mapping objects, profiles, or workflows that you can import into the Model repository for use in the Developer tool.

To import a Marketplace solution, click the Import button next to a Marketplace solution and follow the steps to import the solution into the Model repository. You must be connected to the Model repository to import a solution. You must select a folder during the import process to copy the related source files and documentation for the solution.

After you import the solution into the Model repository, you can run the mapping or you can edit it before you run it.

You can also post a solution to help other users in the Marketplace community.

Setting Up Informatica Developer

To set up the Developer tool, you must establish a connection to the Model repository and select the default Data Integration Service. To establish a connection to the Model repository, you must add the domain and Model repository in the Developer tool. The setup process is different based on what you installed.

If you installed the Informatica services and Informatica client separately, complete the following tasks to set up the Developer tool:

1. Add the domain and Model repository.
2. Select the default Data Integration Service.

If you installed the Informatica services and client at the same time on the same machine, you do not have to set up the Developer tool. The installation process adds the domain, Model repository, and default Data Integration Service in the Developer tool.

After you set up the Developer tool, you can create projects and folders in the Model repository to store your work.
Adding the Domain and Repository

If you installed the Informatica services and Informatica client separately, you must add the domain and Model repository the first time you set up the Developer tool.

1. From the Developer tool menu, click File > Connect to Repository.
   The Connect to Repository dialog box appears.

2. Click Configure Domains.
   The Preferences dialog box appears.

3. Click Add.
   The New Domain dialog box appears.

4. Enter the domain name, host name, and port number for the domain.
The following table lists the default values for the domain:

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain name</td>
<td>Domain_&lt;machine name&gt;</td>
<td>Domain_caw175917</td>
</tr>
<tr>
<td>Host name</td>
<td>&lt;machine name&gt;</td>
<td>caw175917</td>
</tr>
<tr>
<td>Port number</td>
<td>7006</td>
<td>NA</td>
</tr>
</tbody>
</table>

5. Click **Test Connection** to test the connection.
The Developer tool displays a message that states whether the connection is successful.

6. Click **Finish**.
The domain appears in the **Available Domains** panel of the **Preferences** dialog box.

7. Click **OK**.
The **Connect to Repository** dialog box appears.

8. Click **Browse** to select a Model repository.
The **Choose Service** dialog box appears.

9. Expand the domain and select the Model repository.
The following figure shows the selected Model repository:

![Choose Service Window](image)

10. Click **OK** and then click **Next**.

The **Login** window appears.

11. Enter the user name and password provided to you.
12. Click **Finish**.
   The Developer tool connects to the Model repository. The Model repository includes a Samples project that includes sample objects.

Selecting the Default Data Integration Service

If you installed the Informatica services and Informatica client separately, you must select the default Data Integration Service that the Developer tool uses to preview data and run mappings.

1. From the Developer tool menu, click **Window > Preferences**.
   The **Preferences** dialog box appears.
2. Select **Informatica > Data Integration Services**.
3. Expand the domain.
4. Select the Data Integration Service and click **Set as Default**.
The following figure shows the selected default Data Integration Service:

5. Click **OK**.

The Model Repository

The Model repository is a relational database that stores the metadata for projects and folders. If you installed the Informatica client only, you need to add a Model repository when you set up the Developer tool. Each time you open the Developer tool, you connect to the Model repository to access projects and folders.

Objects in Informatica Developer

You can create, manage, or view certain objects in a project or folder in the Developer tool. You can create the following Model repository objects in the Developer tool:

**Application**

- A deployable object that can contain data objects, mappings, and workflows. You can create, edit, and delete applications.

**Folder**

- A container for objects in the Model repository. Use folders to organize objects in a project and create folders to group objects based on business needs. You can create, edit, and delete folders.
Logical data object
An object in a logical data object model that describes a logical entity in an enterprise. It has attributes and keys, and it describes relationships between attributes. You can create, edit, and delete logical data objects in a logical data object model.

Logical data object mapping
A mapping that links a logical data object to one or more physical data objects. It can include transformation logic. You can create, edit, and delete logical data object mappings for a logical data object.

Logical data object model
A data model that contains logical data objects and defines relationships between them. You can create, edit, and delete logical data object models.

Mapping
A set of inputs and outputs linked by transformation objects that define the rules for data transformation. You can create, edit, and delete mappings.

Mapplet
A reusable object that contains a set of transformations that you can use in multiple mappings or validate as a rule. You can create, edit, and delete mapplets.

Physical data object
A physical representation of data that is used to read from, look up, or write to resources. You can create, edit, and delete physical data objects.

Profile
An object that contains rules to discover patterns in source data. Run a profile to evaluate the data structure and verify that data columns contain the type of information that you expect. You can create, edit, and delete profiles.

Rule
Business logic that defines conditions applied to source data when you run a profile. It is a midstream mapplet that you use in a profile. You can create, edit, and delete rules.

Transformation
A repository object in a mapping that generates, modifies, or passes data. Each transformation performs a different function. A transformation can be reusable or non-reusable. You can create, edit, and delete transformations.

Workflow
A graphical representation of a set of events, tasks, and decisions that define a business process. You can create, edit, and delete workflows.

Object Properties
You can view the properties of a project, folder, or any other object in the Model repository.

The General view of the Properties dialog box shows the object properties. Object properties include the name, description, and location of the object in the repository. Object properties also include the user who created and last updated the object and the time the event occurred.

To access the object properties, select the object in the Object Explorer view and click File > Properties.
Connecting to a Model Repository

Each time you open the Developer tool, you connect to a Model repository to access projects and folders. When you connect to a Model repository, you enter connection information to access the domain that includes the Model Repository Service that manages the Model repository.

1. In the Object Explorer view, right-click a Model repository and click Connect.
   The Connect to Repository dialog box appears.
2. Enter the domain user name and password.
3. Click OK.
   The Developer tool connects to the Model repository. The Developer tool displays the projects in the repository.

Projects

A project is the top-level container that you use to store folders and objects in the Developer tool. Use projects to organize and manage the objects that you want to use for data integration solutions.

You manage and view projects in the Object Explorer view. When you create a project, the Developer tool stores the project in the Model repository.

The following table describes the tasks that you can perform on a project:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage projects</td>
<td>Manage project contents. You can create, duplicate, rename, and delete a project. You can view project contents.</td>
</tr>
<tr>
<td>Filter projects</td>
<td>Filter the list of projects that appear in the Object Explorer view.</td>
</tr>
<tr>
<td>Manage folders</td>
<td>Organize project contents in folders. You can create, duplicate, rename, move, and rename folders within projects.</td>
</tr>
<tr>
<td>Manage objects</td>
<td>View object contents, duplicate, rename, move, and delete objects in a project or in a folder within a project.</td>
</tr>
<tr>
<td>Search projects</td>
<td>Search for folders or objects in projects. You can view search results and select an object from the results to view its contents.</td>
</tr>
<tr>
<td>Assign permissions</td>
<td>Depending on your license, you can select the users and groups that can view and edit objects in the project. Specify which users and groups can assign permissions to other users and groups.</td>
</tr>
</tbody>
</table>

Creating a Project

Create a project to store objects and folders.

1. Select a Model Repository Service in the Object Explorer view.
2. Click File > New > Project.
   The New Project dialog box appears.
3. Enter a name for the project.
   If you installed PowerCenter Express Personal Edition, skip to step 6.

4. Click Next.
   The Project Permissions page of the New Project dialog box appears.

5. Optionally, select a user or group and assign permissions.

6. Click Finish.
   The project appears under the Model Repository Service in the Object Explorer view.

Filter Projects

You can filter the list of projects that appear in the Object Explorer view. You might want to filter projects if you have access to a large number of projects but need to manage only some of them.

The Developer tool retains the list of projects that you filter the next time that you connect to the repository.

You can filter projects at the following times:

Before you connect to the repository

When you filter projects before you connect to the repository, you can decrease the amount of time that the Developer tool takes to connect to the repository.

Select File > Connect to Repository. After you select the repository and enter your user name and password, click Next. The Open Project dialog box displays all projects to which you have access. Select the projects that you want to open in the repository and then click Finish.

After you connect to the repository

If you are connected to the repository, click File > Close Projects to filter projects out of the Object Explorer view. The Close Project dialog box displays all projects that are currently open in the Object Explorer view. Select the projects that you want to filter out and then click Finish.

To open projects that you filtered, click File > Open Projects.

Project Permissions

Assign project permissions to users or groups. Project permissions determine whether a user or group can view objects, edit objects, or assign permissions to others.

Depending on the type of PowerCenter Express license, you can assign the following permissions:

Read

The user or group can open, preview, export, validate, and deploy all objects in the project. The user or group can also view project details.

Write

The user or group has read permission on all objects in the project. Additionally, the user or group can edit all objects in the project, edit project details, delete all objects in the project, and delete the project.

Grant

The user or group has read permission on all objects in the project. Additionally, the user or group can assign permissions to other users or groups.
Users assigned the Administrator role for a Model Repository Service inherit all permissions on all projects in the Model Repository Service. Users assigned to a group inherit the group permissions.

Permissions for External Objects

Permissions apply to objects within a project. The Developer tool does not extend permissions to dependent objects when the dependent objects exist in other projects.

Dependent objects are objects that are used by other objects. For example, you create a mapplet that contains a nonreusable Expression transformation. The mapplet is the parent object. The Expression transformation is a dependent object of the mapplet.

The Developer tool creates instances of objects when you use reusable objects within a parent object. For example, you create a mapping with a reusable Lookup transformation. The mapping is the parent object. It contains an instance of the Lookup transformation.

An object can contain instances of dependent objects that exist in other projects. To view dependent object instances from other projects, you must have read permission on the other projects. To edit dependent object instances from other projects, you must have write permission on the parent object project and read permission on the other projects.

Permissions for Dependent Object Instances

You might need to access an object that contains dependent object instances from another project. If you do not have read permission on the other project, the Developer tool gives you different options based on how you access the parent object.

When you try to access a parent object that contains dependent object instances that you cannot view, the Developer tool displays a warning message. If you continue the operation, the Developer tool produces results that vary by operation type.

The following table lists the results of the operations that you can perform on the parent object:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
</table>
| Open the parent object.                        | The Developer tool prompts you to determine how to open the parent object:  
- Open a Copy. The Developer tool creates a copy of the parent object. The copy does not contain the dependent object instances that you cannot view.  
- Open. The Developer tool opens the object, but it removes the dependent object instances that you cannot view. If you save the parent object, the Developer tool removes the dependent object instances from the parent object. The Developer tool does not remove the dependent objects from the repository.  
- Cancel. The Developer tool does not open the parent object.  |
| Export the parent object to an XML file for use in the Developer tool. | The Developer tool creates the export file without the dependent object instances.  |
| Export the parent object to PowerCenter.       | You cannot export the parent object.  |
| Validate the parent object.                   | The Developer tool validates the parent object as if the dependent objects were not part of the parent object.  |
| Deploy the parent object.                     | You cannot deploy the parent object.  |
| Copy and paste the parent object.             | The Developer tool creates the new object without the dependent object instances.  |
Security Details

When you access an object that contains dependent object instances that you cannot view, the Developer tool displays a warning message. The warning message allows you to view details about the dependent objects.

To view details about the dependent objects, click the Details button in the warning message. If you have the Show Security Details Model Repository Service privilege, the Developer tool lists the projects that contain the objects that you cannot view. If you do not have the Show Security Details privilege, the Developer tool indicates that you do not have sufficient privileges to view the project names.

Parent Object Access

If you create parent objects that use dependent object instances from other projects, users might not be able to edit the parent objects. If you want users to be able to edit the parent object and preserve the parent object functionality, you can create instances of the dependent objects in a mapplet.

For example, you create a mapping that contains a reusable Lookup transformation from another project. You want the users of your project to be able to edit the mapping, but not the Lookup transformation.

If you place the Lookup transformation in the mapping, users that do not have read permission on the other project get a warning message when they open the mapping. They can open a copy of the mapping or open the mapping, but the Developer tool removes the Lookup transformation instance.

To allow users to edit the mapping, perform the following tasks:

1. Create a mapplet in your project. Add an Input transformation, the reusable Lookup transformation, and an Output transformation to the mapplet.
2. Edit the mapping, and replace the Lookup transformation with the mapplet.
3. Save the mapping.

When users of your project open the mapping, they see the mapplet instead of the Lookup transformation. The users can edit any part of the mapping except the mapplet.

If users export the mapping, the Developer tool does not include the Lookup transformation in the export file.

Assigning Permissions

You can add users and groups to a project and assign permissions for the users and groups. Assign permissions to determine the tasks that users can complete on objects in the project.

Note: If you have PowerCenter Express Personal Edition, you cannot assign permissions.

1. Select a project in the Object Explorer view.
2. Click File > Properties.
   The Properties window appears.
3. Select Permissions.
4. Click Add to add a user and assign permissions for the user.
   The Domain Users and Groups dialog box appears.
5. To filter the list of users and groups, enter a name or string.
   Optionally, use the wildcard characters in the filter.
6. To filter by security domain, click the Filter by Security Domains button.
7. Select Native to show users and groups in the native security domain. Or, select All to show all users and groups.
8. Select a user or group, and click **OK**.
The user or group appears in the **Project Permissions** page of the **New Project** dialog box.

9. Select read, write, or grant permission for the user or group.
10. Click **OK**.

---

**Folders**

Use folders to organize objects in a project. Create folders to group objects based on business needs. For example, you can create a folder to group objects for a particular task in a project. You can create a folder in a project or in another folder.

Folders appear within projects in the **Object Explorer** view. A folder can contain other folders, data objects, and object types.

You can perform the following tasks on a folder:
- Create a folder.
- View a folder.
- Rename a folder.
- Duplicate a folder.
- Move a folder.
- Delete a folder.

**Creating a Folder**

Create a folder to store related objects in a project. You must create the folder in a project or another folder.

1. In the **Object Explorer** view, select the project or folder where you want to create a folder.
2. Click **File > New > Folder**.
   The **New Folder** dialog box appears.
3. Enter a name for the folder.
4. Click **Finish**.
   The folder appears under the project or parent folder.

---

**Search**

You can search for objects and object properties in the Developer tool.

You can create a search query and then filter the search results. You can view search results and select an object from the results to view its contents. Search results appear on the **Search** view. The search cannot display results if more than 2048 objects are found. If search fails because the results contain more than 2048 objects, change the search options so that fewer objects match the search criteria.
You can use the following search options:

<table>
<thead>
<tr>
<th>Search Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containing text</td>
<td>Object or property that you want to search for. Enter an exact string or use a wildcard. Not case sensitive.</td>
</tr>
<tr>
<td>Name patterns</td>
<td>One or more objects that contain the name pattern. Enter an exact string or use a wildcard. Not case sensitive.</td>
</tr>
<tr>
<td>Search for</td>
<td>One or more object types to search for.</td>
</tr>
<tr>
<td>Scope</td>
<td>Search the workspace or an object that you selected.</td>
</tr>
</tbody>
</table>

The Model Repository Service uses a search engine to index the metadata in the Model repository. To correctly index the metadata, the search engine uses a search analyzer appropriate for the language of the metadata that you are indexing. The Developer tool uses the search engine to perform searches on objects contained in projects in the Model repository. You must save an object before you can search on it.

### Searching for Objects and Properties

Search for objects and properties in the Model repository.

1. Click **Search > Search**.
   
The *Search* dialog box appears.
2. Enter the object or property you want to search for. Optionally, include wildcard characters.
3. If you want to search for a property in an object, optionally enter one or more name patterns separated by a comma.
4. Optionally, choose the object types you want to search for.
5. Choose to search the workspace or the object you selected.
6. Click **Search**.
   
The search results appear in the *Search* view.
7. In the *Search* view, double-click an object to open it in the editor.

### Workspace Editor

Use the editor to view and edit Model repository objects.

You can configure the following arrangement, layout, and navigation options in the editor:

**Align All to Grid**

Arranges objects in the editor based on data flow and aligns them to a grid. Objects retain their original size. You can use this option in a mapping or workflow editor. Open the **Layout** menu to select this option.

**Arrange All**

Aligns the objects in the editor and retains their original order and size. Open the **Layout** menu to select this option.
**Arrange All Iconic**

Converts the objects to icons and aligns the icons in the editor. You can use this option in a mapping or mapplet editor. Open the **Layout** menu to select this option.

**Iconized View**

Reduces objects to named icons. You can view iconized objects in a mapping or mapplet editor.

**Maximize Active View or Editor**

Expands the active window or editor to fill the screen. Click **Window > Navigation** to select this option.

**Minimize Active View or Editor**

Hides the active window or editor. Click **Window > Navigation** to select this option.

**Normal View**

Displays the information in each object in columns. The Developer tool displays objects in the normal view by default.

**Reset Perspective**

Restores all default views and editors. Open the **Window** menu to select this option.

**Resize**

After you resize an object, aligns objects in the editor and retains their current order and size. You can use this option in a mapping or mapplet editor. Hold the **Shift** key while resizing an object to use this option.

**Find in Editor**

Use the editor to find objects, ports, groups, expressions, and attributes that are open in the editor. You can find objects in any mapping, mapplet, logical data object model, or workflow editor. The Developer tool highlights the objects within the open editor.

When you find objects, the Developer tool finds objects that are open in the editor. The objects do not need to be in the Model repository.

To display the find fields below the editor, select **Edit > Find/Replace**. To find an object, specify a search string and the types of objects to find. The types of objects that you can find varies by editor. If you do not specify any type of object, the Developer tool finds the search string in transformations.

When you search for ports, columns, or attributes, you can also select the datatype. For example, you can find integer or bigint ports with names that contain the string "_ID."

The following table lists the types of objects that you can find in each editor:

<table>
<thead>
<tr>
<th>Editor</th>
<th>Object types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping</td>
<td>Mapping objects, expressions, groups, and ports</td>
</tr>
<tr>
<td>Mapplet</td>
<td>Mapplet objects, expressions, groups, and ports</td>
</tr>
<tr>
<td>Logical data object model</td>
<td>Logical data objects and attributes</td>
</tr>
<tr>
<td>Physical data object read or write mapping</td>
<td>Mapping objects and columns</td>
</tr>
<tr>
<td>Workflow</td>
<td>Workflow objects</td>
</tr>
</tbody>
</table>
When the Developer tool finds the search string, it displays the object locations. It also highlights the object in which the search string occurs. If the search string occurs in an iconized transformation in the mapping editor, the Developer tool highlights the iconized transformation.

You can select the following options to navigate the results of a find:

- **Next Match.** Finds the next occurrence of the search string.
- **Previous Match.** Finds the previous occurrence of the search string.
- **Highlight All.** Highlights all occurrences of the search string.
- **Expand Iconized Transformations.** Expands all iconized transformations in which the search string occurs.

### Validation Preferences

You can limit the number of error messages that appear in the Validation Log view. You can also group error messages by object or object type in the Validation Log view.

#### Grouping Error Messages

Group error messages in the Validation Log view to organize messages by object or object type. Otherwise, messages appear alphabetically.

To group error messages in the Validation Log view, select **Menu > Group By** and then select **Object** or **Object Type**.

To remove error message groups, select **Menu > Group By > None**. Error messages appear ungrouped, listed alphabetically in the Validation Log view.

#### Limiting Error Messages

You can limit the number of error messages that appear in the Validation Log view. The limit determines how many messages appear in a group or the total number of messages that appear in the Validation Log view. Error messages are listed alphabetically and get deleted from bottom to top when a limit is applied.

1. Click **Window > Preferences**.
   The Preferences dialog box appears.
2. Select **Informatica > Validation**.
3. Optionally, set the error limit and configure the number of items that appear.
   Default is 100.
4. To restore the default values, click **Restore Defaults**.
5. Click **Apply**.
6. Click **OK**.
Copy

You can copy objects within a project or to a different project. You can also copy objects to folders in the same project or to folders in a different project.

You can copy the following objects to another project or folder or save copies of the objects with different names:

- Application
- Logical data object model
- Mapping
- Mapplet
- Physical data object
- Profile
- Reusable transformation
- Rule
- Workflow

Use the following guidelines when you copy objects:

- You can copy segments of mappings, mapplets, and rules.
- You can copy a folder to another project.
- You can paste an object multiple times after you copy it.
- If the project or folder contains an object with the same name, you can rename or replace the object.

Copying an Object

Copy an object to make it available in another project or folder.

1. Select an object in a project or folder.
2. Click **Edit > Copy**.
3. Select the project or folder that you want to copy the object to.
4. Click **Edit > Paste**.

Saving a Copy of an Object

Save a copy of an object to save the object with a different name.

1. Open an object in the editor.
2. Click **File > Save a Copy As**.
3. Enter a name for the copy of the object.
4. Click **Browse** to select the project or folder that you want to copy the object to.
5. Click **Finish**.
Tags

A tag is metadata that defines an object in the Model repository based on business usage. Create tags to group objects according to their business usage.

After you create a tag, you can associate the tag with one or more objects. You can remove the association between a tag and an object. You can use a tag to search for objects associated with the tag in the Model repository. The Developer tool displays a glossary of all tags.

For example, you create a tag named XYZCorp_CustomerOrders and assign it to tables that contain information for the customer orders from the XYZ Corporation. Users can search by the XYZCorp_CustomerOrders tag to identify the tables associated with the tag.

Creating a Tag

Create a tag to add metadata that defines an object based on business usage.

1. To create a tag, use one of the following methods:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preferences dialog box</td>
<td>Click Window &gt; Preferences. In the Preferences dialog box, select Informatica &gt; Tags. Select a Model Repository Service and click Add.</td>
</tr>
<tr>
<td>Tags view for an object</td>
<td>Open an object in the editor. In the Tags view, click Edit. In the Assign Tags for Object dialog box, click New.</td>
</tr>
</tbody>
</table>

2. Enter a name for the tag.
3. Optionally, enter a description.
4. Click OK.

Assigning a Tag

Assign a tag to an object to associate the object with the metadata definition.

1. Open an object in the editor.
2. In the Tags view, click Edit.
   - The Assign Tags for Object dialog box appears. The Available Tags area displays all tags defined in the repository. You can search for a tag by name or description. The Assign Tags area displays the opened object and any tags assigned to the object.
3. In the Available Tags area, select a tag.
4. In the Assign Tags area, select the object.
5. Click Assign.
6. To remove a tag from an object, select the tag in the Available Tags area and the object in the Assign Tags area, and then click Remove.

Viewing Tags

You can view all tags assigned to an object, or you can view all tags defined in the Model repository.

1. To view tags assigned to an object, open the object in the editor.
2. Select the **Tags** view.
   The **Tags** view displays all tags assigned to the object.

3. To view all the tags defined in the Model repository, click **Window > Preferences**.
   The **Preferences** dialog box appears.

4. Select **Informatica > Tags**.
   The **Tags** area displays all tags defined in the Model repository. You can search for a tag by name or description.
Connections Overview

A connection is a repository object that defines a connection in the domain configuration repository. Create a connection to import data objects, preview data, profile data, and run mappings. The Developer tool uses the connection when you import a data object. The Data Integration Service uses the connection when you preview data, run mappings, or consume web services.

The Developer tool stores connections in the domain configuration repository. Any connection that you create in the Developer tool is available in the Administrator tool.

Create and manage connections using the Create Connection and Show Connections buttons.
The following figure shows the **Create Connection** and the **Show Connections** buttons in the Developer tool:

![Image](image-url)

1. Create Connection
2. Show Connections
3. Create Connection - Connection Explorer view

After you create a connection, you can perform the following actions on the connection:

**Edit the connection.**

You can change the connection name and the description. You can also edit connection details such as the user name, password, and connection strings.

The Data Integration Service identifies connections by the connection ID. Therefore, you can change the connection name. When you rename a connection, the Developer tool updates the objects that use the connection.

Parameter files identify a connection by name, not by connection ID. Therefore, when you rename a connection, you must also update all parameter files that use the connection parameter.

**Copy the connection.**

Copy a connection to create a connection that is similar to another connection. For example, you might create two Oracle connections that differ only in user name and password.

**Delete the connection.**

When you delete a connection, objects that use the connection are no longer valid. If you accidentally delete a connection, you can re-create it by creating another connection with the same connection ID as the deleted connection.

**Refresh the connections list.**

You can refresh the connections list to see the latest list of connections for the domain. Refresh the connections list after a user adds, deletes, or renames a connection in the Administrator tool.

---

**Connection Explorer View**

Use the **Connection Explorer** view to view relational or nonrelational database connections and to create relational or nonrelational data objects.
The following figure shows the **Connection Explorer** view in the Developer tool:

![Connection Explorer](image)

You can complete the following tasks in the **Connection Explorer** view:

- Create a database connection using the **Create Connection** button.
- Add a connection to the view. Click the **Select Connection** button to choose one or more connections to add to the **Connection Explorer** view.
- Connect to a relational or nonrelational database. Right-click the database and click **Connect**.
- Disconnect from a relational or nonrelational database. Right-click the database and click **Disconnect**.
- Create a relational data object. After you connect to a relational database, expand the database to view tables. Right-click a table, and click **Add to Project** to open the **New Relational Data Object** dialog box.
- Create a nonrelational data object. After you connect to a nonrelational database, expand the database to view data maps. Right-click a data map, and click **Add to Project** to open the **New Non-relational Data Object** dialog box.
- Refresh a connection. Right-click a connection and click **Refresh**.
- Show only the default schema. Right-click a connection and click **Show Default Schema Only**. Default is enabled.
- Delete a connection from the **Connection Explorer** view. The connection remains in the Model repository. Right-click a connection and click **Delete**.

**IBM DB2 Connection Properties**

Use an IBM DB2 connection to access tables in an IBM DB2 database.

The following table describes the IBM DB2 connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>Database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>Connection String for metadata access</td>
<td>Connection string to import physical data objects. Use the following connection string: <code>jdbc:informatica:db2://&lt;host&gt;:&lt;port&gt;/&lt;dbname&gt;</code></td>
</tr>
<tr>
<td>Connection String for data access</td>
<td>Connection string to preview data and run mappings. Enter <code>dbname</code> from the alias configured in the DB2 client.</td>
</tr>
</tbody>
</table>
### JDBC Connection Properties

Use a JDBC connection to access tables in a database through JDBC.

The following table describes the JDBC connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>Database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>JDBC Driver Class Name</td>
<td>Name of the JDBC driver class.</td>
</tr>
<tr>
<td>Connection String</td>
<td>Connection string to connect to the database. Use the following connection string: <code>jdbc:&lt;subprotocol&gt;;&lt;subname&gt;</code></td>
</tr>
<tr>
<td>Environment SQL</td>
<td>Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the connection environment SQL each time it connects to the database.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Transaction SQL</td>
<td>Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the transaction environment SQL at the beginning of each transaction.</td>
</tr>
<tr>
<td>SQL Identifier Character</td>
<td>The type of character used to identify special characters and reserved SQL keywords, such as WHERE. The Data Integration Service places the selected character around special characters and reserved SQL keywords. The Data Integration Service also uses this character for the Support Mixed-case Identifiers property.</td>
</tr>
<tr>
<td>Support Mixed-case Identifiers</td>
<td>When enabled, the Data Integration Service places identifier characters around table, view, schema, synonym, and column names when generating and executing SQL against these objects in the connection. Use if the objects have mixed-case or lowercase names. By default, this option is not selected.</td>
</tr>
</tbody>
</table>

Microsoft SQL Server Connection Properties

Use a Microsoft SQL Server connection to access tables in a Microsoft SQL Server database.

The following table describes the Microsoft SQL Server connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>Database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>Use Trusted Connection</td>
<td>Optional. When enabled, the Data Integration Service uses Windows authentication to access the Microsoft SQL Server database. The user name that starts the Data Integration Service must be a valid Windows user with access to the Microsoft SQL Server database.</td>
</tr>
<tr>
<td>Connection String for metadata access</td>
<td>Connection string to import physical data objects. Use the following connection string: jdbc:informatica:sqlserver://&lt;host&gt;:&lt;port&gt;;&lt;dbname&gt;</td>
</tr>
<tr>
<td>Connection String for data access</td>
<td>Connection string to preview data and run mappings. Enter &lt;ServerName&gt;@&lt;DBName&gt;</td>
</tr>
<tr>
<td>Domain Name</td>
<td>Optional. Name of the domain where Microsoft SQL Server is running.</td>
</tr>
<tr>
<td>Packet Size</td>
<td>Required. Optimize the ODBC connection to Microsoft SQL Server. Increase the packet size to increase performance. Default is 0.</td>
</tr>
<tr>
<td>Code Page</td>
<td>Database code page.</td>
</tr>
<tr>
<td>Owner Name</td>
<td>Name of the schema owner. Specify for connections to the profiling warehouse database, staging database, or data object cache database.</td>
</tr>
<tr>
<td>Schema Name</td>
<td>Name of the schema in the database. Specify for connections to the profiling warehouse database, staging database, or data object cache database. You must specify the</td>
</tr>
</tbody>
</table>
**Property** | **Description**
--- | ---
User name | Database user name.
Password | Password for the user name.
Connection String | Connection string to connect to the database.
Code Page | Database code page.
Environment SQL | Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the connection environment SQL each time it connects to the database.

**Note:** When you use a Microsoft SQL Server connection to access tables in a Microsoft SQL Server database, the Developer tool does not display the synonyms for the tables.

**ODBC Connection Properties**

Use an ODBC connection to access tables in a database through ODBC.

The following table describes the ODBC connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>Database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>Connection String</td>
<td>Connection string to connect to the database.</td>
</tr>
<tr>
<td>Code Page</td>
<td>Database code page.</td>
</tr>
<tr>
<td>Environment SQL</td>
<td>Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the connection environment SQL each time it connects to the database.</td>
</tr>
</tbody>
</table>
### Oracle Connection Properties

Use an Oracle connection to access tables in an Oracle database.

The following table describes the Oracle connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>Database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>Connection String for metadata access</td>
<td>Connection string to import physical data objects. Use the following connection string: <code>jdbc:informatica:oracle://&lt;host&gt;:1521;SID=&lt;sid&gt;</code></td>
</tr>
<tr>
<td>Connection String for data access</td>
<td>Connection string to preview data and run mappings. Enter <code>dbname.world</code> from the TNSNAMES entry.</td>
</tr>
<tr>
<td>Code Page</td>
<td>Database code page.</td>
</tr>
<tr>
<td>Environment SQL</td>
<td>Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the connection environment SQL each time it connects to the database.</td>
</tr>
<tr>
<td>Transaction SQL</td>
<td>Optional. Enter SQL commands to set the database environment when you connect to the database. The Data Integration Service executes the transaction environment SQL at the beginning of each transaction.</td>
</tr>
</tbody>
</table>
### Property Description

**Retry Period**
Number of seconds the Data Integration Service attempts to reconnect to the database if the connection fails. If the Data Integration Service cannot connect to the database in the retry period, the session fails. Default is 0.

**Parallel Mode**
Optional. Enables parallel processing when loading data into a table in bulk mode. Default is disabled.

**SQL Identifier Character**
The type of character used to identify special characters and reserved SQL keywords, such as WHERE. The Data Integration Service places the selected character around special characters and reserved SQL keywords. The Data Integration Service also uses this character for the Support Mixed-case Identifiers property.

**Support Mixed-case Identifiers**
When enabled, the Data Integration Service places identifier characters around table, view, schema, synonym, and column names when generating and executing SQL against these objects in the connection. Use if the objects have mixed-case or lowercase names. By default, this option is not selected.

### Web Services Connection Properties

Use a web services connection to connect a Web Service Consumer transformation to a web service.

The following table describes the web services connection properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username</strong></td>
<td>User name to connect to the web service. Enter a user name if you enable HTTP authentication or WS-Security. If the Web Service Consumer transformation includes WS-Security ports, the transformation receives a dynamic user name through an input port. The Data Integration Service overrides the user name defined in the connection.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Password for the user name. Enter a password if you enable HTTP authentication or WS-Security. If the Web Service Consumer transformation includes WS-Security ports, the transformation receives a dynamic password through an input port. The Data Integration Service overrides the password defined in the connection.</td>
</tr>
<tr>
<td><strong>End Point URL</strong></td>
<td>URL for the web service that you want to access. The Data Integration Service overrides the URL defined in the WSDL file. If the Web Service Consumer transformation includes an endpoint URL port, the transformation dynamically receives the URL through an input port. The Data Integration Service overrides the URL defined in the connection.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Number of seconds that the Data Integration Service waits for a response from the web service provider before it closes the connection.</td>
</tr>
</tbody>
</table>
## Property | Description
--- | ---
**HTTP Authentication Type** | Type of user authentication over HTTP. Select one of the following values:
- **None. No authentication.**
- **Automatic.** The Data Integration Service chooses the authentication type of the web service provider.
- **Basic.** Requires you to provide a user name and password for the domain of the web service provider. The Data Integration Service sends the user name and the password to the web service provider for authentication.
- **Digest.** Requires you to provide a user name and password for the domain of the web service provider. The Data Integration Service generates an encrypted message digest from the user name and password and sends it to the web service provider. The provider generates a temporary value for the user name and password and stores it in the Active Directory on the Domain Controller. It compares the value with the message digest. If they match, the web service provider authenticates you.
- **NTLM.** Requires you to provide a domain name, server name, or default user name and password. The web service provider authenticates you based on the domain you are connected to. It gets the user name and password from the Windows Domain Controller and compares it with the user name and password that you provide. If they match, the web service provider authenticates you. NTLM authentication does not store encrypted passwords in the Active Directory on the Domain Controller.

**WS Security Type** | Type of WS-Security that you want to use. Select one of the following values:
- **None.** The Data Integration Service does not add a web service security header to the generated SOAP request.
- **PasswordText.** The Data Integration Service adds a web service security header to the generated SOAP request. The password is stored in the clear text format.
- **PasswordDigest.** The Data Integration Service adds a web service security header to the generated SOAP request. The password is stored in a digest form which provides effective protection against replay attacks over the network. The Data Integration Service combines the password with a nonce and a time stamp. The Data Integration Service applies a SHA hash on the password, encodes it in base64 encoding, and uses the encoded password in the SOAP header.

**Trust Certificates File** | File containing the bundle of trusted certificates that the Data Integration Service uses when authenticating the SSL certificate of the web service. Enter the file name and full directory path. Default is `<Informatica installation directory>/services/shared/bin/ca-bundle.crt`.

**Client Certificate File Name** | Client certificate that a web service uses when authenticating a client. Specify the client certificate file if the web service needs to authenticate the Data Integration Service.

**Client Certificate Password** | Password for the client certificate. Specify the client certificate file if the web service needs to authenticate the Data Integration Service.

**Client Certificate Type** | Format of the client certificate file. Select one of the following values:
- **PEM.** Files with the .pem extension.
- **DER.** Files with the .cer or .der extension.

**Private Key File Name** | Private key file for the client certificate. Specify the private key file if the web service needs to authenticate the Data Integration Service.

**Private Key Password** | Password for the private key of the client certificate. Specify the private key password if the web service needs to authenticate the Data Integration Service.

**Private Key Type** | Type of the private key. PEM is the supported type.

---

**Connection Management**

Create and manage connections in the **Preferences** dialog box or the **Connection Explorer** view.
Creating Connections

Create a database, social media, or web services connection. Create the connection before you import physical data objects, preview data, profile data, or run mappings. You can also add a database connection to the Connection Explorer view after you create it.

1. Click Create Connection button in the Developer tool bar.
   The New <Connection Type> Connection dialog box appears.
2. Select the type of connection that you want to create.
   - To select a database connection, select Databases.
   - To select a social media connection, select Social Media.
   - To select a web services connection, select Web Services Connections.
3. Enter the following information:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the connection. The name is not case sensitive and must be unique within the domain. It cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ ^ &amp; * ( ) - + = [ ]</td>
</tr>
<tr>
<td>ID</td>
<td>String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional description for the connection.</td>
</tr>
<tr>
<td>Location</td>
<td>Domain in which the connection exists.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of connection.</td>
</tr>
</tbody>
</table>
4. Click Next.
5. Configure the connection properties.
6. Click Test Connection to verify that you entered the connection properties correctly and that you can connect.
7. Click Finish.
8. When you create a database connection, the Add Connection dialog box appears. You can choose to add the connection to the Connection Explorer view.
   The new connection appears in the Connection Explorer view.

Showing Connections

You can see and manage available connections in the Informatica domain.

1. Click Show Connections in the Developer toolbar.
   The Preferences dialog box appears.
2. Select the type of connection.
   - To select a non-web services connection, select Informatica > Connections.
   - To select a web services connection, select Informatica > Web Services > Connections.
3. Expand the domain in the Available Connections list to see available connections.
   You can also add, edit, remove, copy, and refresh the connection list.

**Refreshing the Connections List**

Refresh the connections list to see the latest list of connections in the domain.

1. Click Window > Preferences.
2. Select the type of connection that you want to refresh.
   - To select a non-web services connection, select Informatica > Connections.
   - To select a web services connection, select Informatica > Web Services > Connections.
3. Select the domain in the Available Connections list.
4. Click Refresh.
5. Expand the domain in the Available Connections list to see the latest list of connections.
6. Click OK to close the Preferences dialog box.

**Editing a Connection**

You can edit the connection name, description, and connection properties.

1. Click Window > Preferences.
2. Select the type of connection that you want to edit.
   - To select a non-web services connection, select Informatica > Connections.
   - To select a web services connection, select Informatica > Web Services > Connections.
3. Expand the domain in the Available Connections list.
4. Select the connection in Available Connections, and click Edit.
   The Edit Connection dialog box appears.
5. Optionally, edit the connection name and description.
   **Note:** If you change a connection name, you must redeploy all applications that use the connection. You must also update all parameter files that use the connection parameter.
6. Click Next.
7. Optionally, edit the connection properties.
8. Click Test Connection to verify that you entered the connection properties correctly and that you can connect to the database.
9. Click OK to close the Edit Connection dialog box.
10. Click OK to close the Preferences dialog box.

**Copying a Connection**

You can copy a connection within a domain or into another domain.

1. Click Window > Preferences.
2. Select the type of connection that you want to copy.
   - To select a non-web services connection, select Informatica > Connections.
   - To select a web services connection, select Informatica > Web Services > Connections.
3. Expand the domain in the Available Connections list.
4. Select the connection in Available Connections, and click Copy.
   The Copy Connection dialog box appears.
5. Enter the connection name and ID, and select the domain.
   The name and ID must be unique in the domain.
6. Click OK to close the Copy Connection dialog box.
7. Click OK to close the Preferences dialog box.

Deleting a Connection

When you delete a connection through the Preferences dialog box, the Developer tool removes the connection from the Model repository.

1. Click Window > Preferences.
2. Select the type of connection that you want to delete.
   ♦ To select a non-web services connection, select Informatica > Connections.
   ♦ To select a web services connection, select Informatica > Web Services > Connections.
3. Expand the domain in the Available Connections list.
4. Select the connection in Available Connections, and click Remove.
5. Click OK to close the Preferences dialog box.
Physical Data Objects

This chapter includes the following topics:

- Physical Data Objects Overview, 39
- Relational Data Objects, 40
- Customized Data Objects, 43
- Custom Queries, 48
- Flat File Data Objects, 59
- WSDL Data Object, 71
- Synchronization, 75
- Troubleshooting Physical Data Objects, 76

Physical Data Objects Overview

A physical data object is the physical representation of data that is used to read from, look up, or write to resources.

A physical data object can be one of the following types:

Relational data object

A physical data object that uses a relational table, view, or synonym as a source. For example, you can create a relational data object from an Oracle view.

Depending on the object type, you can add a relational data object to a mapping or mapplet as a source, a target, or a Lookup transformation.

Customized data object

A physical data object that uses one or multiple related relational resources or relational data objects as sources. Relational resources include tables, views, and synonyms. For example, you can create a customized data object from two Microsoft SQL Server tables that have a primary key-foreign key relationship.

Create a customized data object if you want to perform operations such as joining data, filtering rows, sorting ports, or running custom queries in a reusable data object.

Flat file data object

A physical data object that uses a flat file as a source. You can create a flat file data object from a delimited or fixed-width flat file.
WSDL data object

A physical data object that uses a WSDL file as a source.

If the data object source changes, you can synchronize the physical data object. When you synchronize a physical data object, the Developer tool reimports the object metadata.

You can create any physical data object in a project or folder. Physical data objects in projects and folders are reusable objects. You can use them in any type of mapping, mapplet, or profile, but you cannot change the data object within the mapping, mapplet, or profile. To update the physical data object, you must edit the object within the project or folder.

You can include a physical data object in a mapping, mapplet, or profile. You can add a physical data object to a mapping or mapplet as a read, write, or lookup transformation. You can add a physical data object to a logical data object mapping to map logical data objects.

Relational Data Objects

Import a relational data object to include in a mapping, mapplet, or profile. A relational data object is a physical data object that uses a relational table, view, or synonym as a source.

The following figure shows a sample relational data object that is open in the editor:

You can create primary key-foreign key relationships between relational data objects. You can create key relationships between relational data objects whether or not the relationships exist in the source database.

You can include relational data objects in mappings and mapplets. You can add relational data object to a mapping or mapplet as a read, write, or lookup transformation. You can add multiple relational data objects to a mapping or mapplet as sources. When you add multiple relational data objects at the same time, the Developer tool prompts you to add the objects in either of the following ways:

- As related data objects. The Developer tool creates one read transformation. The read transformation has the same capabilities as a customized data object.
As independent data objects. The Developer tool creates one read transformation for each relational data object. The read transformations have the same capabilities as relational data objects.

You can import the following types of relational data object:
- IBM DB2
- JDBC
- Microsoft SQL Server
- ODBC
- Oracle

Key Relationships

You can create key relationships between relational data objects. Key relationships allow you to join relational data objects when you use them as sources in a customized data object or as read transformations in a mapping or mapplet.

When you import relational data objects, the Developer tool retains the primary key information defined in the database. When you import related relational data objects at the same time, the Developer tool also retains foreign keys and key relationships. However, if you import related relational data objects separately, you must re-create the key relationships after you import the objects.

To create key relationships between relational data objects, first create a primary key in the referenced object. Then create the relationship in the relational data object that contains the foreign key.

The key relationships that you create exist in the relational data object metadata. You do not need to alter the source relational resources.

Creating Keys in a Relational Data Object

Create key columns to identify each row in a relational data object. You can create one primary key in each relational data object.

1. Open the relational data object.
2. Select the Keys view.
   The following figure shows the Keys view for a sample relational data object that is open in the editor:

3. Click Add.
   The New Key dialog box appears.
4. Enter a key name.
5. If the key is a primary key, select **Primary Key**.
6. Select the key columns.
7. Click **OK**.
8. Save the relational data object.

**Creating Relationships between Relational Data Objects**

You can create key relationships between relational data objects. You cannot create key relationships between a relational data object and a customized data object.

The relational data object that you reference must have a primary key.

1. Open the relational data object where you want to create a foreign key.
2. Select the **Relationships** view.
3. Click **Add**.
   
   The **New Relationship** dialog box appears.
4. Enter a name for the foreign key.
5. Select a primary key from the referenced relational data object.
6. Click **OK**.
7. In the **Relationships** properties, select the foreign key columns.
8. Save the relational data object.

**Creating a Read Transformation from Relational Data Objects**

You can a add relational data object to a mapping or mapplet as a read transformation. When you add multiple relational data objects at the same time, you can add them as related or independent objects.

1. Open the mapping or mapplet in which you want to create a read transformation.
2. In the **Object Explorer** view, select one or more relational data objects.
3. Drag the relational data objects into the mapping editor.
   
   The **Add to Mapping** dialog box appears.
4. Select the **Read** option.
5. If you add multiple data objects, select one of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>As related data objects</td>
<td>The Developer tool creates one read transformation. The read transformation has the same capabilities as a customized data object.</td>
</tr>
<tr>
<td>As independent data objects</td>
<td>The Developer tool creates one read transformation for each relational data object. Each read transformation has the same capabilities as a relational data object.</td>
</tr>
</tbody>
</table>

6. If the relational data objects use different connections, select the default connection.
7. Click **OK**.
   
   The Developer tool creates one or multiple read transformations in the mapping or mapplet.
Importing a Relational Data Object

Import a relational data object to add to a mapping, mapplet, or profile.

Before you import a relational data object, you must configure a connection to the database.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File > New > Data Object**.
   The **New** dialog box appears.
3. Select **Relational Data Object** and click **Next**.
   The **New Relational Data Object** dialog box appears.
4. Click **Browse** next to the **Connection** option and select a connection to the database.
5. Click **Create data object from existing resource**.
6. Click **Browse** next to the **Resource** option and select the table, view, or synonym that you want to import.
7. Enter a name for the physical data object.
8. Click **Browse** next to the **Location** option and select the project where you want to import the relational data object.
9. Click **Finish**.
   The data object appears under **Physical Data Objects** in the project or folder in the **Object Explorer** view.

Customized Data Objects

Customized data objects are reusable physical data objects with one or more relational resources. Create a customized data object if you want to perform operations such as joining data, filtering rows, sorting ports, or running custom queries when the Data Integration Service reads source data. You can reuse a customized data object in a mapping, mapplet, or profile.

You can create customized data objects in projects and folders. You cannot change the customized data object from within a mapping, mapplet, or profile. If you change a customized data object in a project or folder, the Developer tool updates the object in all mappings, mapplets, and profiles that use the object.
The following figure shows a sample customized data object that is open in the editor:

Create a customized data object to perform the following tasks:

- Create a custom query to replace the default query that the Data Integration Service runs to read the source data. The default query is a SELECT statement that references each column that the Data Integration Service reads from the source.

- Define parameters for the data object. You can define and assign parameters in a customized data object to represent connections. When you run a mapping that uses the customized data object, you can define different values for the connection parameters at runtime.

- Join source data that originates from the same source database. You can join multiple tables with primary key-foreign key relationships whether or not the relationships exist in the database.

- Retain key relationships when you synchronize the object with the sources. If you create a customized data object that contains multiple tables, and you define key relationships that do not exist in the database, you can retain the relationships when you synchronize the data object.

- Select distinct values from the source. If you choose Select Distinct, the Data Integration Service adds a SELECT DISTINCT statement to the default SQL query.

- Filter rows when the Data Integration Service reads source data. If you include a filter condition, the Data Integration Service adds a WHERE clause to the default query.

- Specify sorted ports. If you specify a number for sorted ports, the Data Integration Service adds an ORDER BY clause to the default SQL query.

- Specify an outer join instead of the default inner join. If you include a user-defined join, the Data Integration Service replaces the join information specified by the metadata in the SQL query.

- Add pre- and post-mapping SQL commands. The Data Integration Service runs pre-mapping SQL commands against the source database before it reads the source. It runs post-mapping SQL commands against the source database after it writes to the target.

You can create customized data objects from the following types of connections and objects:

- IBM DB2 connections
You can also add sources to a customized data object through a custom SQL query.

Key Relationships

You can create key relationships between sources in a customized data object when the sources are relational resources. Key relationships allow you to join the sources within the customized data object.

**Note:** If a customized data object uses relational data objects as sources, you cannot create key relationships within the customized data object. You must create key relationships between the relational data objects instead.

When you import relational resources into a customized data object, the Developer tool retains the primary key information defined in the database. When you import related relational resources into a customized data object at the same time, the Developer tool also retains key relationship information. However, if you import related relational resources separately, you must re-create the key relationships after you import the objects into the customized data object.

When key relationships exist between sources in a customized data object, the Data Integration Service joins the sources based on the related keys in each source. The default join is an inner equijoin that uses the following syntax in the WHERE clause:

```
Source1.column_name = Source2.column_name
```

You can override the default join by entering a user-defined join or by creating a custom query.

To create key relationships in a customized data object, first create a primary key in the referenced source transformation. Then create the relationship in the source transformation that contains the foreign key.

The key relationships that you create exist in the customized data object metadata. You do not need to alter the source relational resources.

Customized Data Object Write Properties

The Data Integration Service uses write properties when it writes data to relational resources. To edit write properties, select the Input transformation in the Write view, and then select the Advanced properties.

The following table describes the write properties that you configure for customized data objects:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load type</td>
<td>Type of target loading. Select Normal or Bulk.</td>
</tr>
<tr>
<td></td>
<td>If you select Normal, the Data Integration Service loads targets normally. You can choose Bulk when you load to DB2, Sybase, Oracle, or Microsoft SQL Server. If you specify Bulk for other database types, the Data Integration Service reverts to a normal load. Bulk loading can increase mapping performance, but it limits the ability to recover because no database logging occurs. Choose Normal mode if the mapping contains an Update Strategy transformation. If you choose Normal and the Microsoft SQL Server target name includes spaces, configure the following environment SQL in the connection object: SET QUOTE_IDENTIFIER ON</td>
</tr>
<tr>
<td>Update override</td>
<td>Overrides the default UPDATE statement for the target.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes all rows flagged for delete. Default is enabled.</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts all rows flagged for insert. Default is enabled.</td>
</tr>
<tr>
<td>Truncate target table</td>
<td>Truncates the target before it loads data. Default is disabled.</td>
</tr>
<tr>
<td>Update strategy</td>
<td>Update strategy for existing rows. You can select one of the following strategies:</td>
</tr>
<tr>
<td></td>
<td>- Update as update. The Data Integration Service updates all rows flagged for update.</td>
</tr>
<tr>
<td></td>
<td>- Update as insert. The Data Integration Service inserts all rows flagged for update. You must also select the Insert target option.</td>
</tr>
<tr>
<td></td>
<td>- Update else insert. The Data Integration Service updates rows flagged for update if they exist in the target and then inserts any remaining rows marked for insert. You must also select the Insert target option.</td>
</tr>
<tr>
<td>PreSQL</td>
<td>SQL command the Data Integration Service runs against the target database before it reads the source. The Developer tool does not validate the SQL.</td>
</tr>
<tr>
<td>PostSQL</td>
<td>SQL command the Data Integration Service runs against the target database after it writes to the target. The Developer tool does not validate the SQL.</td>
</tr>
</tbody>
</table>

Creating a Customized Data Object

Create a customized data object to add to a mapping, mapplet, or profile. After you create a customized data object, add sources to it.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File > New > Data Object**.
   The **New** dialog box appears.
3. Select **Relational Data Object** and click **Next**.
   The **New Relational Data Object** dialog box appears.
4. Click **Browse** next to the Connection option and select a connection to the database.
5. Click **Create customized data object**.
6. Enter a name for the customized data object.
7. Click **Browse** next to the Location option and select the project where you want to create the customized data object.
8. Click **Finish**.
   The customized data object appears under Physical Data Objects in the project or folder in the **Object Explorer** view.

Add sources to the customized data object. You can add relational resources or relational data objects as sources. You can also use a custom SQL query to add sources.

Adding Relational Resources to a Customized Data Object

After you create a customized data object, add sources to it. You can use relational resources as sources.
Before you add relational resources to a customized data object, you must configure a connection to the database.

1. In the Connection Explorer view, select one or more relational resources in the same relational connection.
2. Right-click in the Connection Explorer view and select Add to project.
   The Add to Project dialog box appears.
3. Select Add as related resource(s) to existing customized data object and click OK.
   The Add to Data Object dialog box appears.
4. Select the customized data object and click OK.
5. If you add multiple resources to the customized data object, the Developer tool prompts you to select the resource to write to. Select the resource and click OK.
   If you use the customized data object in a mapping as a write transformation, the Developer tool writes data to this resource.
   The Developer tool adds the resources to the customized data object.

Adding Relational Data Objects to a Customized Data Object

After you create a customized data object, add sources to it. You can use relational data objects as sources.

1. Open the customized data object.
2. Select the Read view.
3. In the Object Explorer view, select one or more relational data objects in the same relational connection.
4. Drag the objects from the Object Explorer view to the customized data object Read view.
5. If you add multiple relational data objects to the customized data object, the Developer tool prompts you to select the object to write to. Select the object and click OK.
   If you use the customized data object in a mapping as a write transformation, the Developer tool writes data to this relational data object.
   The Developer tool adds the relational data objects to the customized data object.

Creating Keys in a Customized Data Object

Create key columns to identify each row in a source transformation. You can create one primary key in each source transformation.

1. Open the customized data object.
2. Select the Read view.
3. Select the source transformation where you want to create a key.
   The source must be a relational resource, not a relational data object. If the source is a relational data object, you must create keys in the relational data object.
4. Select the Keys properties.
5. Click Add.
   The New Key dialog box appears.
6. Enter a key name.
7. If the key is a primary key, select Primary Key.
8. Select the key columns.
9. Click OK.
Creating Relationships within a Customized Data Object

You can create key relationships between sources in a customized data object.

The source transformation that you reference must have a primary key.

1. Open the customized data object.
2. Select the Read view.
3. Select the source transformation where you want to create a foreign key.
   - The source must be a relational resource, not a relational data object. If the source is a relational data object, you must create relationships in the relational data object.
4. Select the Relationships properties.
5. Click Add.
   - The New Relationship dialog box appears.
6. Enter a name for the foreign key.
7. Select a primary key from the referenced source transformation.
8. Click OK.
9. In the Relationships properties, select the foreign key columns.
10. Save the customized data object.

Custom Queries

A custom SQL query is a SELECT statement that overrides the default SQL query in a customized data object or relational data object. When you define a custom query in a customized data object, you can reuse the object in multiple mappings or profiles. When you define the query in a relational data object, you must define it for an instance of the relational data object that is in a mapping, mapplet, or profile.

A custom query overrides the default SQL query that the Data Integration Service uses to read data from the source relational resource. The custom query also overrides the simple query settings you define when you enter a source filter, use sorted ports, enter a user-defined join, or select distinct ports.

Use the following guidelines when you create a custom query in a customized data object or relational data object:

- In the SELECT statement, list the column names in the order in which they appear in the source transformation.
- Enclose all database reserved words in quotes.

If you use a customized data object to perform a self-join, you must enter a custom SQL query that includes the self-join.

You can use a customized data object with a custom query as a read transformation in a mapping. The source database executes the query before it passes data to the Data Integration Service.

You can create a custom query to add sources to an empty customized data object. You can also use a custom query to override the default SQL query.
Creating a Custom Query

Create a custom query to issue a special SELECT statement for reading data from the sources. The custom query overrides the default query that the Data Integration Service issues to read source data.

1. Open the customized data object or the relational data object instance.
2. Select the Read view.
3. Select the Output transformation.
4. Select the Query properties.
5. Select the advanced query.
6. Select Use custom query.

The Data Integration Service displays the query it issues to read source data.

7. Change the query or replace it with a custom query.

The following figure shows a sample custom query:

8. Save the data object.

Default Query

The Data Integration Service generates a default SQL query that it uses to read data from relational sources. You can override the default query in a customized data object or an instance of a relational data object.

You can override the default query through the simple or advanced query. Use the simple query to select distinct values, enter a source filter, sort ports, or enter a user-defined join. Use the advanced query to create a custom SQL query for reading data from the sources. The custom query overrides the default and simple queries.

If any table name or column name contains a database reserved word, you can create and maintain a reserved words file, reswords.txt. Create the reswords.txt file on any machine the Data Integration Service can access.

When the Data Integration Service runs a mapping, it searches for the reswords.txt file. If the file exists, the Data Integration Service places quotes around matching reserved words when it executes SQL against the database. If you override the default query, you must enclose any database reserved words in quotes.

When the Data Integration Service generates the default query, it delimits table and field names containing the following characters with double quotes:

/ + - ~ ` ! @ ^ & * ( ) [ ] { } ; : ; , < > \ | <space>
Creating a Reserved Words File

Create a reserved words file if any table name or column name in the customized data object contains a database reserved word.

You must have administrator privileges to configure the Data Integration Service to use the reserved words file.

1. Create a file called "reswords.txt."
2. Create a section for each database by entering the database name within square brackets, for example, [Oracle].
3. Add the reserved words to the file below the database name.
   
   For example:
   
   ```
   [Oracle]
   OPTION
   START
   where
   number
   [SQL Server]
   CURRENT
   where
   number
   ```

   Entries are not case sensitive.
4. Save the reswords.txt file.
5. In Informatica Administrator, select the Data Integration Service.
6. Edit the custom properties.
7. Add the following custom property:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved Words File</td>
<td>&lt;path&gt;/reswords.txt</td>
</tr>
</tbody>
</table>

8. Restart the Data Integration Service.

Hints

You can add hints to the source SQL query to pass instructions to a database optimizer. The optimizer uses the hints to choose a query run plan to access the source.

The Hints field appears in the Query view of a relational data object instance or a customized data object. The source database must be Oracle, Sybase, IBM DB2, or Microsoft SQL Server. The Hints field does not appear for other database types.

When the Data Integration Service generates the source query, it adds the SQL hints to the query exactly as you enter them in the Developer tool. The Data Integration Service does not parse the hints. When you run the mapping that contains the source, the mapping log shows the query with the hints in the query.

The Data Integration Service inserts the SQL hints in a position in the query depending on the database type. Refer to your database documentation for information about the syntax for hints.

Oracle

The Data Integration Service add hints directly after the SELECT/UPDATE/INSERT/DELETE keyword.

```
SELECT /*+ <hints> */ FROM ...
```

The '+' indicates the start of hints.

The hints are contained in a comment (/* ... */ or --... until end of line)
Sybase
The Data Integration Service adds hints after the query. Configure a plan name in the hint.

```sql
SELECT ... PLAN <plan>
select avg(price) from titles plan "(scalar_agg i_scan type_price_ix titles )"
```

IBM DB2
You can enter the optimize-for clause as a hint. The Data Integration Service adds the clause at the end of the query.

```sql
SELECT ... OPTIMIZE FOR <n> ROWS
```

The optimize-for clause tells the database optimizer how many rows the query might process. The clause does not limit the number of rows. If the database processes more than <n> rows, then performance might decrease.

Microsoft SQL Server
The Data Integration Service adds hints at the end of the query as part of an OPTION clause.

```sql
SELECT ... OPTION { <query_hints> }
```

Hints Rules and Guidelines
Use the following rules and guidelines when you configure hints for SQL queries:

- If you enable pushdown optimization or if you use a semi-join in a relational data object, then the original source query changes. The Data Integration Service does not apply hints to the modified query.
- You can combine hints with join and filter overrides, but if you configure a SQL override, the SQL override takes precedence and the Data Integration Service does not apply the other overrides.
- The Query view shows a simple view or an advanced view. If you enter a hint with a filter, sort, or join override on the simple view, and you the Developer tool shows the full query override on the advanced view.

Creating Hints
Create hints to send instructions to the database optimizer to determine a query plan.

1. Open the customized data object or the relational data object instance.
2. Select the Read view.
3. Select the Output transformation.
4. Select the Query properties.
5. Select the simple query.
6. Click Edit next to the Hints field.
   The Hints dialog box appears.
7. Enter the hint in the SQL Query field.
   The Developer tool does not validate the hint.
8. Click OK.
9. Save the data object.
Select Distinct

You can select unique values from sources in a customized data object or a relational data object instance with the select distinct option. When you enable select distinct, the Data Integration Service adds a SELECT DISTINCT statement to the default SQL query.

Use the select distinct option to filter source data. For example, you might use the select distinct option to extract unique customer IDs from a table that lists total sales. When you use the relational data object in a mapping, the Data Integration Service filters data earlier in the data flow, which can increase performance.

Using Select Distinct

Select unique values from a relational source with the **Select Distinct** property.

1. Open the customized data object or relational data object instance.
2. Select the **Read** view.
3. Select the Output transformation.
4. Select the **Query** properties.
5. Select the simple query.
6. Enable the **Select Distinct** option.
7. Save the customized data object.

Filters

You can enter a filter value in a custom query. The filter becomes the WHERE clause in the query SELECT statement. Use a filter to reduce the number of rows that the Data Integration Service reads from the source table.

Entering a Source Filter

Enter a source filter to reduce the number of rows the Data Integration Service reads from the relational source.

1. Open the customized data object or the relational data object instance.
2. Select the **Read** view.
3. Select the Output transformation.
4. Select the **Query** properties.
5. Select the simple query.
6. Click **Edit** next to the **Filter** field.
   
   The **SQL Query** dialog box appears.
7. Enter the filter condition in the **SQL Query** field.
   
   You can select column names from the **Columns** list.
8. Click **OK**.
9. Click **Validate** to validate the filter condition.
10. Save the data object.
Sorted Ports

You can sort rows in the default query for a customized data object or a relational data object instance. Select the ports to sort by. The Data Integration Service adds the ports to the ORDER BY clause in the default query.

You might sort the source rows to increase performance when you include the following transformations in a mapping:

- Aggregator. When you configure an Aggregator transformation for sorted input, you can send sorted data by using sorted ports. The group by ports in the Aggregator transformation must match the order of the sorted ports in the customized data object.
- Joiner. When you configure a Joiner transformation for sorted input, you can send sorted data by using sorted ports. Configure the order of the sorted ports the same in each customized data object.

Note: You can also use the Sorter transformation to sort relational and flat file data before Aggregator and Joiner transformations.

Sorting Column Data

Use sorted ports to sort column data in a customized data object or relational data object instance. When you use the data object as a read transformation in a mapping or mapplet, you can pass the sorted data to transformations downstream from the read transformation.

1. Open the customized data object or relational data object instance.
2. Select the Read view.

The following figure shows the Read view of a customized data object that is open in the editor:

3. Select the Output transformation.
4. Select the Query properties.
5. Select the simple query.
6. Click Edit next to the Sort field.

The Sort dialog box appears.
7. To specify a column as a sorted port, click the New button.
8. Select the column and sort type, either ascending or descending.
9. Repeat steps Using Sorted Ports and Using Sorted Ports to select other columns to sort.

The Developer tool sorts the columns in the order in which they appear in the Sort dialog box.
10. Click OK.
In the **Query** properties, the Developer tool displays the sort columns in the **Sort** field.

11. Click **Validate** to validate the sort syntax.
12. Save the data object.

### User-Defined Joins

You can configure a user-defined join in a customized data object or relational data object instance. A user-defined join defines the condition to join data from multiple sources in the same data object.

When you add a user-defined join to a customized data object or a relational data object instance, you can use the data object as a read transformation in a mapping. The source database performs the join before it passes data to the Data Integration Service. Mapping performance increases when the source tables are indexed.

Create a user-defined join to join data from related sources. The user-defined join overrides the default inner join that the Data Integration creates based on the related keys in each source. When you enter a user-defined join, enter the contents of the WHERE clause that specifies the join condition. If the user-defined join performs an outer join, the Data Integration Service might insert the join syntax in the WHERE clause or the FROM clause, based on the database syntax.

You might need to enter a user-defined join in the following circumstances:

- Columns do not have a primary key-foreign key relationship.
- The datatypes of columns used for the join do not match.
- You want to specify a different type of join, such as an outer join.

Use the following guidelines when you enter a user-defined join in a customized data object or relational data object instance:

- Do not include the WHERE keyword in the user-defined join.
- Enclose all database reserved words in quotes.
- If you use Informatica join syntax, and Enable quotes in SQL is enabled for the connection, you must enter quotes around the table names and the column names if you enter them manually. If you select tables and columns when you enter the user-defined join, the Developer tool places quotes around the table names and the column names.

User-defined joins join data from related resources in a database. To join heterogeneous sources, use a Joiner transformation in a mapping that reads data from the sources. To perform a self-join, you must enter a custom SQL query that includes the self-join.

### Entering a User-Defined Join

Configure a user-defined join in a customized data object or relational data object to define the join condition for the data object sources.

1. Open the customized data object or relational data object instance.
2. Select the **Read** view.
3. Select the Output transformation.
4. Select the **Query** properties.
5. Select the simple query.
6. Click **Edit** next to the **Join** field.
   
   The **SQL Query** dialog box appears.
7. Enter the user-defined join in the **SQL Query** field.
You can select column names from the Columns list.

8. Click OK.

9. Click Validate to validate the user-defined join.

10. Save the data object.

Outer Join Support

You can use a customized data object to perform an outer join of two sources in the same database. When the Data Integration Service performs an outer join, it returns all rows from one source resource and rows from the second source resource that match the join condition.

Use an outer join when you want to join two resources and return all rows from one of the resources. For example, you might perform an outer join when you want to join a table of registered customers with a monthly purchases table to determine registered customer activity. You can join the registered customer table with the monthly purchases table and return all rows in the registered customer table, including customers who did not make purchases in the last month. If you perform a normal join, the Data Integration Service returns only registered customers who made purchases during the month, and only purchases made by registered customers.

With an outer join, you can generate the same results as a master outer or detail outer join in the Joiner transformation. However, when you use an outer join, you reduce the number of rows in the data flow which can increase performance.

You can enter two kinds of outer joins:

- Left. The Data Integration Service returns all rows for the resource to the left of the join syntax and the rows from both resources that meet the join condition.
- Right. The Data Integration Service returns all rows for the resource to the right of the join syntax and the rows from both resources that meet the join condition.

Note: Use outer joins in nested query statements when you override the default query.

You can enter an outer join in a user-defined join or in a custom SQL query.

Informatica Join Syntax

When you enter join syntax, use the Informatica or database-specific join syntax. When you use the Informatica join syntax, the Data Integration Service translates the syntax and passes it to the source database during a mapping run.

Note: Always use database-specific syntax for join conditions.

When you use Informatica join syntax, enclose the entire join statement in braces ({Informatica syntax}). When you use database syntax, enter syntax supported by the source database without braces.

When you use Informatica join syntax, use table names to prefix column names. For example, if you have a column named FIRST_NAME in the REG_CUSTOMER table, enter "REG_CUSTOMER.FIRST_NAME" in the join syntax. Also, when you use an alias for a table name, use the alias within the Informatica join syntax to ensure the Data Integration Service recognizes the alias.

You can combine left outer or right outer joins with normal joins in a single data object. You cannot combine left and right outer joins. Use multiple normal joins and multiple left outer joins. Some databases limit you to using one right outer join.

When you combine joins, enter the normal joins first.
Normal Join Syntax

You can create a normal join using the join condition in a customized data object or relational data object instance. When you create an outer join, you must override the default join. As a result, you must include the normal join in the join override. When you include a normal join in the join override, list the normal join before outer joins. You can enter multiple normal joins in the join override.

To create a normal join, use the following syntax:

```
(source1 INNER JOIN source2 on join_condition)
```

The following table displays the syntax for normal joins in a join override:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source1</td>
<td>Source resource name. The Data Integration Service returns rows from this resource that match the join condition.</td>
</tr>
<tr>
<td>source2</td>
<td>Source resource name. The Data Integration Service returns rows from this resource that match the join condition.</td>
</tr>
<tr>
<td>join_condition</td>
<td>Condition for the join. Use syntax supported by the source database. You can combine multiple join conditions with the AND operator.</td>
</tr>
</tbody>
</table>

For example, you have a REG_CUSTOMER table with data for registered customers:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>Marviñ</td>
<td>Chi</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
</tr>
<tr>
<td>00003</td>
<td>John</td>
<td>Bowden</td>
</tr>
<tr>
<td>00004</td>
<td>J.</td>
<td>Marks</td>
</tr>
</tbody>
</table>

The PURCHASES table, refreshed monthly, contains the following data:

<table>
<thead>
<tr>
<th>TRANSACTION_NO</th>
<th>CUST_ID</th>
<th>DATE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>06-2000-0001</td>
<td>00001</td>
<td>6/3/2000</td>
<td>55.79</td>
</tr>
<tr>
<td>06-2000-0002</td>
<td>00002</td>
<td>6/10/2000</td>
<td>104.45</td>
</tr>
<tr>
<td>06-2000-0003</td>
<td>00001</td>
<td>6/10/2000</td>
<td>255.56</td>
</tr>
<tr>
<td>06-2000-0004</td>
<td>00004</td>
<td>6/15/2000</td>
<td>534.95</td>
</tr>
<tr>
<td>06-2000-0005</td>
<td>00002</td>
<td>6/21/2000</td>
<td>98.65</td>
</tr>
<tr>
<td>06-2000-0007</td>
<td>NULL</td>
<td>6/24/2000</td>
<td>325.45</td>
</tr>
</tbody>
</table>

To return rows displaying customer names for each transaction in the month of June, use the following syntax:

```
(REG_CUSTOMER INNER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID)
```

The Data Integration Service returns the following data:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>DATE</th>
<th>AMOUNT</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>00002</td>
<td>6/3/2000</td>
<td>55.79</td>
<td>Dinah</td>
<td>Jones</td>
</tr>
<tr>
<td>00002</td>
<td>6/10/2000</td>
<td>104.45</td>
<td>Dinah</td>
<td>Jones</td>
</tr>
<tr>
<td>00001</td>
<td>6/10/2000</td>
<td>255.56</td>
<td>Marvin</td>
<td>Chi</td>
</tr>
<tr>
<td>00004</td>
<td>6/15/2000</td>
<td>534.95</td>
<td>J.</td>
<td>Marks</td>
</tr>
<tr>
<td>00002</td>
<td>6/21/2000</td>
<td>98.65</td>
<td>Dinah</td>
<td>Jones</td>
</tr>
</tbody>
</table>

The Data Integration Service returns rows with matching customer IDs. It does not include customers who made no purchases in June. It also does not include purchases made by non-registered customers.

Left Outer Join Syntax

You can create a left outer join with a join override. You can enter multiple left outer joins in a single join override. When using left outer joins with other joins, list all left outer joins together, after any normal joins in the statement.
To create a left outer join, use the following syntax:

```
( source1 LEFT OUTER JOIN source2 on join_condition )
```

The following tables display syntax for left outer joins in a join override:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source1</td>
<td>Source resource name. With a left outer join, the Data Integration Service returns all rows in this resource.</td>
</tr>
<tr>
<td>source2</td>
<td>Source resource name. The Data Integration Service returns rows from this resource that match the join condition.</td>
</tr>
<tr>
<td>join_condition</td>
<td>Condition for the join. Use syntax supported by the source database. You can combine multiple join conditions with the AND operator.</td>
</tr>
</tbody>
</table>

For example, using the same REG_CUSTOMER and PURCHASES tables described in “Normal Join Syntax” on page 56, you can determine how many customers bought something in June with the following join override:

```
( REG_CUSTOMER LEFT OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID )
```

The Data Integration Service returns the following data:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>DATE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>Marvin</td>
<td>Chi</td>
<td>6/10/2000</td>
<td>255.56</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/3/2000</td>
<td>55.79</td>
</tr>
<tr>
<td>00003</td>
<td>John</td>
<td>Bowden</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00004</td>
<td>J.</td>
<td>Marks</td>
<td>6/10/2000</td>
<td>104.45</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/10/2000</td>
<td>104.45</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/21/2000</td>
<td>96.65</td>
</tr>
</tbody>
</table>

The Data Integration Service returns all registered customers in the REG_CUSTOMERS table, using null values for the customer who made no purchases in June. It does not include purchases made by non-registered customers.

Use multiple join conditions to determine how many registered customers spent more than $100.00 in a single purchase in June:

```
( REG_CUSTOMER LEFT OUTER JOIN PURCHASES on (REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID AND PURCHASES.AMOUNT > 100.00) )
```

The Data Integration Service returns the following data:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>DATE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>Marvin</td>
<td>Chi</td>
<td>6/10/2000</td>
<td>255.56</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/3/2000</td>
<td>55.79</td>
</tr>
<tr>
<td>00003</td>
<td>John</td>
<td>Bowden</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00004</td>
<td>J.</td>
<td>Marks</td>
<td>6/15/2000</td>
<td>534.95</td>
</tr>
</tbody>
</table>

You might use multiple left outer joins if you want to incorporate information about returns during the same time period. For example, the RETURNS table contains the following data:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>CUST_ID</th>
<th>RETURN</th>
</tr>
</thead>
<tbody>
<tr>
<td>00002</td>
<td>6/10/2000</td>
<td>55.79</td>
</tr>
<tr>
<td>00002</td>
<td>6/21/2000</td>
<td>104.45</td>
</tr>
</tbody>
</table>

To determine how many customers made purchases and returns for the month of June, use two left outer joins:

```
( REG_CUSTOMER LEFT OUTER JOIN PURCHASES on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID LEFT OUTER JOIN RETURNS on REG_CUSTOMER.CUST_ID = PURCHASES.CUST_ID )
```

The Data Integration Service returns the following data:

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>DATE</th>
<th>AMOUNT</th>
<th>RET_DATE</th>
<th>RETURN</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>Marvin</td>
<td>Chi</td>
<td>6/10/2000</td>
<td>255.56</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/3/2000</td>
<td>55.79</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00003</td>
<td>John</td>
<td>Bowden</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00004</td>
<td>J.</td>
<td>Marks</td>
<td>6/15/2000</td>
<td>534.95</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/10/2000</td>
<td>104.45</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>00002</td>
<td>Dinah</td>
<td>Jones</td>
<td>6/21/2000</td>
<td>96.65</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
The Data Integration Service uses NULLs for missing values.

**Right Outer Join Syntax**

You can create a right outer join with a join override. The right outer join returns the same results as a left outer join if you reverse the order of the resources in the join syntax. Use only one right outer join in a join override. If you want to create more than one right outer join, try reversing the order of the source resources and changing the join types to left outer joins.

When you use a right outer join with other joins, enter the right outer join at the end of the join override.

To create a right outer join, use the following syntax:

```
{ source1 RIGHT OUTER JOIN source2 on join_condition }
```

The following table displays syntax for a right outer join in a join override:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>source1</code></td>
<td>Source resource name. The Data Integration Service returns rows from this resource that match</td>
</tr>
<tr>
<td></td>
<td>the join condition.</td>
</tr>
<tr>
<td><code>source2</code></td>
<td>Source resource name. With a right outer join, the Data Integration Service returns all rows</td>
</tr>
<tr>
<td></td>
<td>in this resource.</td>
</tr>
<tr>
<td><code>join_condition</code></td>
<td>Condition for the join. Use syntax supported by the source database. You can combine multiple</td>
</tr>
<tr>
<td></td>
<td>join conditions with the AND operator.</td>
</tr>
</tbody>
</table>

**Pre- and Post-Mapping SQL Commands**

You can create SQL commands in a customized data object or relational data object instance. The Data Integration Service runs the SQL commands against the source relational resource.

When you run the mapping, the Data Integration Service runs pre-mapping SQL commands against the source database before it reads the source. It runs post-mapping SQL commands against the source database after it writes to the target.

Use the following guidelines when you configure pre- and post-mapping SQL commands:

- Use any command that is valid for the database type. The Data Integration Service does not allow nested comments, even though the database might allow them.
- Use a semicolon (;) to separate multiple statements. The Data Integration Service issues a commit after each statement.
- The Data Integration Service ignores semicolons within /* ... */.
- If you need to use a semicolon outside comments, you can escape it with a backslash (\). When you escape the semicolon, the Data Integration Service ignores the backslash, and it does not use the semicolon as a statement separator.
- The Developer tool does not validate the SQL in a pre- and post-mapping SQL commands.

**Adding Pre- and Post-Mapping SQL Commands**

You can add pre- and post-mapping SQL commands to a customized data object or relational data object instance. The Data Integration Service runs the SQL commands when you use the data object in a mapping.
1. Open the customized data object.
2. Select the Read view.
3. Select the Output transformation
4. Select the Advanced properties.
5. Enter a pre-mapping SQL command in the PreSQL field.
6. Enter a post-mapping SQL command in the PostSQL field.
7. Save the customized data object.

**Flat File Data Objects**

Create or import a flat file data object to include in a mapping, mapplet, or profile. You can use flat file data objects as sources, targets, and lookups in mappings and mapplets. You can create profiles on flat file data objects.

A flat file physical data object can be delimited or fixed-width. You can import fixed-width and delimited flat files that do not contain binary data.

After you import a flat file data object, you might need to create parameters or configure file properties. Create parameters through the Parameters view. Edit file properties through the Overview, Read, Write, and Advanced views.

The Overview view allows you to edit the flat file data object name and description. It also allows you to update column properties for the flat file data object.

The Read view controls the properties that the Data Integration Service uses when it reads data from the flat file. The Read view contains the following transformations:

- Source transformation. Defines the flat file that provides the source data. Select the source transformation to edit properties such as the name and description, column properties, and source file format properties.
- Output transformation. Represents the rows that the Data Integration Service reads when it runs a mapping. Select the Output transformation to edit the file run-time properties such as the source file name and directory.

The Write view controls the properties that the Data Integration Service uses when it writes data to the flat file. The Write view contains the following transformations:

- Input transformation. Represents the rows that the Data Integration Service writes when it runs a mapping. Select the Input transformation to edit the file run-time properties such as the target file name and directory.
- Target transformation. Defines the flat file that accepts the target data. Select the target transformation to edit the name and description and the target file format properties.

The Advanced view controls format properties that the Data Integration Service uses when it reads data from and writes data to the flat file.

When you create mappings that use file sources or file targets, you can view flat file properties in the Properties view. You cannot edit file properties within a mapping, except for the reject file name, reject file directory, and tracing level.

**Flat File Data Object Overview Properties**

The Data Integration Service uses overview properties when it reads data from or writes data to a flat file. Overview properties include general properties, which apply to the flat file data object. They also include column properties, which apply to the columns in the flat file data object. The Developer tool displays overview properties for flat files in the Overview view.
The following table describes the general properties that you configure for flat files:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the flat file data object.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the flat file data object.</td>
</tr>
</tbody>
</table>

The following table describes the column properties that you configure for flat files:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the column.</td>
</tr>
<tr>
<td>Native type</td>
<td>Native datatype of the column.</td>
</tr>
<tr>
<td>Bytes to process (fixed-width flat files)</td>
<td>Number of bytes that the Data Integration Service reads or writes for the column.</td>
</tr>
<tr>
<td>Precision</td>
<td>Maximum number of significant digits for numeric datatypes, or maximum number of characters for string datatypes. For numeric datatypes, precision includes scale.</td>
</tr>
<tr>
<td>Scale</td>
<td>Maximum number of digits after the decimal point for numeric values.</td>
</tr>
<tr>
<td>Format</td>
<td>Column format for numeric and datetime datatypes.</td>
</tr>
<tr>
<td></td>
<td>For numeric datatypes, the format defines the thousand separator and decimal separator. Default is no thousand separator and a period (.) for the decimal separator.</td>
</tr>
<tr>
<td></td>
<td>For datetime datatypes, the format defines the display format for year, month, day, and time. It also defines the field width. Default is &quot;A 19 YYYY-MM-DD HH24:MI:SS.&quot;</td>
</tr>
<tr>
<td>Visibility</td>
<td>Determines whether the Data Integration Service can read data from or write data to the column. For example, when the visibility is Read, the Data Integration Service can read data from the column. It cannot write data to the column. For flat file data objects, this property is read-only. The visibility is always Read and Write.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the column.</td>
</tr>
</tbody>
</table>

**Flat File Data Object Read Properties**

The Data Integration Service uses read properties when it reads data from a flat file. Select the source transformation to edit general, column, and format properties. Select the Output transformation to edit run-time properties.
The following figure shows the read properties for the selected SALES_TRANSACTIONS source transformation:

1. Source transformation
2. Output transformation
3. Read view
4. Properties view

**General Properties**

The Developer tool displays general properties for flat file sources in the source transformation in the Read view.

The following table describes the general properties that you configure for flat file sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the flat file. This property is read-only. You can edit the name in the Overview view. When you use the flat file as a source in a mapping, you can edit the name within the mapping.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the flat file.</td>
</tr>
</tbody>
</table>
## Column Properties

The Developer tool displays column properties for flat file sources in the source transformation in the Read view.

The following table describes the column properties that you configure for flat file sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the column.</td>
</tr>
<tr>
<td>Native type</td>
<td>Native datatype of the column.</td>
</tr>
<tr>
<td>Bytes to process (fixed-width flat files)</td>
<td>Number of bytes that the Data Integration Service reads for the column.</td>
</tr>
<tr>
<td>Precision</td>
<td>Maximum number of significant digits for numeric datatypes, or maximum number of characters for string datatypes. For numeric datatypes, precision includes scale.</td>
</tr>
<tr>
<td>Scale</td>
<td>Maximum number of digits after the decimal point for numeric values.</td>
</tr>
<tr>
<td>Format</td>
<td>Column format for numeric and datetime datatypes. For numeric datatypes, the format defines the thousand separator and decimal separator. Default is no thousand separator and a period (.) for the decimal separator. For datetime datatypes, the format defines the display format for year, month, day, and time. It also defines the field width. Default is &quot;A 19 YYYY-MM-DD HH24:MI:SS.&quot;</td>
</tr>
<tr>
<td>Shift key (fixed-width flat files)</td>
<td>Allows the user to define a shift-in or shift-out statefulness for the column in the fixed-width flat file.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the column.</td>
</tr>
</tbody>
</table>

## Format Properties

The Developer tool displays format properties for flat file sources in the source transformation in the Read view.

The following table describes the format properties that you configure for delimited flat file sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start import at line</td>
<td>Row at which the Data Integration Service starts importing data. Use this option to skip header rows. Default is 1.</td>
</tr>
<tr>
<td>Row delimiter</td>
<td>Octal code for the character that separates rows of data. Default is line feed, \012 LF (\n).</td>
</tr>
<tr>
<td>Escape character</td>
<td>Character used to escape a delimiter character in an unquoted string if the delimiter is the next character after the escape character. If you specify an escape character, the Data Integration Service reads the delimiter character as a regular character embedded in the string. <strong>Note:</strong> You can improve mapping performance slightly if the source file does not contain quotes or escape characters.</td>
</tr>
<tr>
<td>Retain escape character in data</td>
<td>Includes the escape character in the output string.</td>
</tr>
</tbody>
</table>
The following table describes the format properties that you configure for fixed-width flat file sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start import at line</td>
<td>Row at which the Data Integration Service starts importing data. Use this option to skip header rows. Default is 1.</td>
</tr>
<tr>
<td>Number of bytes to skip between records</td>
<td>Number of bytes between the last column of one row and the first column of the next. The Data Integration Service skips the entered number of bytes at the end of each row to avoid reading carriage return characters or line feed characters. Enter 1 for UNIX files and 2 for DOS files. Default is 2.</td>
</tr>
<tr>
<td>Line sequential</td>
<td>Causes the Data Integration Service to read a line feed character or carriage return character in the last column as the end of the column. Select this option if the file uses line feeds or carriage returns to shorten the last column of each row. Default is disabled.</td>
</tr>
<tr>
<td>Strip trailing blanks</td>
<td>Strips trailing blanks from string values. Default is disabled.</td>
</tr>
<tr>
<td>User defined shift state</td>
<td>Allows you to select the shift state for source columns in the Columns properties. Select this option when the source file contains both multibyte and single-byte data, but does not contain shift-in and shift-out keys. If a multibyte file source does not contain shift keys, you must select a shift key for each column in the flat file data object. Select the shift key for each column to enable the Data Integration Service to read each character correctly. Default is disabled.</td>
</tr>
</tbody>
</table>
Run-time Properties

The Developer tool displays run-time properties for flat file sources in the Output transformation in the Read view.

The following table describes the run-time properties that you configure for flat file sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Input type        | Type of source input. You can choose the following types of source input:  
- File. For flat file sources.  
- Command. For source data or a file list generated by a shell command. |
| Source type       | Indicates source type of files with the same file properties. You can choose one of the following source types:  
- Direct. A source file that contains the source data.  
- Indirect. A source file that contains a list of files. The Data Integration Service reads the file list and reads the files in sequential order.  
- Directory. Source files that are in a directory. You must specify the directory location in the source file directory property. The Data Integration Service reads the files in ascending alphabetic order. The Data Integration Service does not read files in the subdirectories. |
| Source file name  | File name of the flat file source.                                                                                                           |
| Source file directory | Directory where the flat file sources exist. The machine that hosts Informatica services must be able to access this directory.  
Default is the SourceDir system parameter. |
| Command           | Command used to generate the source file data. Use a command to generate or transform flat file data and send the standard output of the command to the flat file reader when the mapping runs.  
The flat file reader reads the standard output as the flat file source data. Generating source data with a command eliminates the need to stage a flat file source. Use a command or script to send source data directly to the Data Integration Service instead of using a pre-mapping command to generate a flat file source. You can also use a command to generate a file list.  
For example, to use a directory listing as a file list, use the following command:  
```bash  
cd MySourceFiles; ls sales-records-Sep-1-2005.dat  
```                                                                                                                                 |
| Truncate string null | Strips the first null character and all characters after the first null character from string values.  
Enable this option for delimited flat files that contain null characters in strings. If you do not enable this option, the Data Integration Service generates a row error for any row that contains null characters in a string.  
Default is disabled. |
| Line sequential buffer length | Number of bytes that the Data Integration Service reads for each line.  
This property, together with the total row size, determines whether the Data Integration Service drops a row. If the row exceeds the larger of the line sequential buffer length or the total row size, the Data Integration Service drops the row and writes it to the mapping log file. To determine the total row size, add the column precision and the delimiters, and then multiply the total by the maximum bytes for each character.  
Default is 1024. |

Configuring Flat File Read Properties

Configure read properties to control how the Data Integration Service reads data from a flat file.

1. Open the flat file data object.
2. Select the Read view.
3. To edit general, column, or format properties, select the source transformation. To edit run-time properties, select the Output transformation.

4. In the Properties view, select the properties you want to edit. For example, click Columns properties or Runtime properties.

5. Edit the properties.

6. Save the flat file data object.

Flat File Data Object Write Properties

The Data Integration Service uses write properties when it writes data to a flat file. Select the Input transformation to edit run-time properties. Select the target transformation to edit general and column properties.

The following figure shows the write properties for the selected Input transformation:
Run-time Properties

The Developer tool displays run-time properties for flat file targets in the Input transformation in the Write view.
The following table describes the run-time properties that you configure for flat file targets:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append if exists</td>
<td>Appends the output data to the target files and reject files. If you do not select this option, the Data Integration Service truncates the target file and reject file before writing data to them. If the files do not exist, the Data Integration Service creates them. Default is disabled.</td>
</tr>
<tr>
<td>Create directory if not exists</td>
<td>Creates the target directory if it does not exist. Default is disabled.</td>
</tr>
<tr>
<td>Header options</td>
<td>Creates a header row in the file target. You can choose the following options: - No header. Does not create a header row in the file target. - Output field names. Creates a header row in the file target with the output port names. - Use header command output. Uses the command in the Header Command field to generate a header row. For example, you can use a command to add the date to a header row for the file target. Default is no header.</td>
</tr>
<tr>
<td>Header command</td>
<td>Command used to generate the header row in the file target.</td>
</tr>
<tr>
<td>Footer command</td>
<td>Command used to generate the footer row in the file target.</td>
</tr>
<tr>
<td>Output type</td>
<td>Type of target for the mapping. Select File to write the target data to a flat file. Select Command to output data to a command.</td>
</tr>
<tr>
<td>Output file directory</td>
<td>Output directory for the flat file target. The machine that hosts Informatica services must be able to access this directory. Default is the TargetDir system parameter.</td>
</tr>
<tr>
<td>Output file name</td>
<td>File name of the flat file target.</td>
</tr>
<tr>
<td>Command</td>
<td>Command used to process the target data. On UNIX, use any valid UNIX command or shell script. On Windows, use any valid DOS command or batch file. The flat file writer sends the data to the command instead of a flat file target. You can improve mapping performance by pushing transformation tasks to the command instead of the Data Integration Service. You can also use a command to sort or to compress target data. For example, use the following command to generate a compressed file from the target data: <code>compress -c &gt; MyTargetFiles/MyCompressedFile.Z</code></td>
</tr>
<tr>
<td>Reject file directory</td>
<td>Directory where the reject file exists. Default is the RejectDir system parameter. <strong>Note:</strong> This field appears when you edit a flat file target in a mapping.</td>
</tr>
<tr>
<td>Reject file name</td>
<td>File name of the reject file. <strong>Note:</strong> This field appears when you edit a flat file target in a mapping.</td>
</tr>
</tbody>
</table>
General Properties
The Developer tool displays general properties for flat file targets in the target transformation in the Write view.
The following table describes the general properties that you configure for flat file targets:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the flat file. This property is read-only. You can edit the name in the Overview view. When you use the flat file as a target in a mapping, you can edit the name within the mapping.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the flat file.</td>
</tr>
</tbody>
</table>

Column Properties
The Developer tool displays column properties for flat file targets in the target transformation in the Write view.
The following table describes the column properties that you configure for flat file targets:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the column.</td>
</tr>
<tr>
<td>Native type</td>
<td>Native datatype of the column.</td>
</tr>
<tr>
<td>Bytes to process (fixed-width flat files)</td>
<td>Number of bytes that the Data Integration Service writes for the column.</td>
</tr>
<tr>
<td>Precision</td>
<td>Maximum number of significant digits for numeric datatypes, or maximum number of characters for string datatypes. For numeric datatypes, precision includes scale.</td>
</tr>
<tr>
<td>Scale</td>
<td>Maximum number of digits after the decimal point for numeric values.</td>
</tr>
<tr>
<td>Format</td>
<td>Column format for numeric and datetime datatypes. For numeric datatypes, the format defines the thousand separators and decimal separators. Default is no thousand separator and a period (.) for the decimal separator. For datetime datatypes, the format defines the display format for year, month, day, and time. It also defines the field width. Default is &quot;A 19 YYYY-MM-DD HH24:MI:SS.&quot;</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the field width.</td>
</tr>
</tbody>
</table>

Configuring Flat File Write Properties
Configure write properties to control how the Data Integration Service writes data to a flat file.

1. Open the flat file data object.
2. Select the Write view.
3. To edit run-time properties, select the Input transformation. To edit general or column properties, select the target transformation.
4. In the Properties view, select the properties you want to edit.
   For example, click Runtime properties or Columns properties.
5. Edit the properties.
6. Save the flat file data object.

Flat File Data Object Advanced Properties

The Data Integration Service uses advanced properties when it reads data from or writes data to a flat file. The Developer tool displays advanced properties for flat files in the Advanced view.

The following table describes the advanced properties that you configure for flat files:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code page</td>
<td>Code page of the flat file data object. For source files, use a source code page that is a subset of the target code page. For lookup files, use a code page that is a superset of the source code page and a subset of the target code page. For target files, use a code page that is a superset of the source code page. Default is &quot;MS Windows Latin 1 (ANSI), superset of Latin 1.&quot;</td>
</tr>
<tr>
<td>Format</td>
<td>Format for the flat file, either delimited or fixed-width.</td>
</tr>
<tr>
<td>Delimiters (delimited flat files)</td>
<td>Character used to separate columns of data.</td>
</tr>
<tr>
<td>Null character type (fixed-width flat files)</td>
<td>Null character type, either text or binary.</td>
</tr>
<tr>
<td>Null character (fixed-width flat files)</td>
<td>Character used to represent a null value. The null character can be any valid character in the file code page or any binary value from 0 to 255.</td>
</tr>
<tr>
<td>Repeat null character (fixed-width flat files)</td>
<td>For source files, causes the Data Integration Service to read repeat null characters in a single field as one null value. For target files, causes the Data Integration Service to write as many null characters as possible into the target field. If you do not enable this option, the Data Integration Service enters one null character at the beginning of the field to represent a null value. Default is disabled.</td>
</tr>
<tr>
<td>Datetime format</td>
<td>Defines the display format and the field width for datetime values. Default is &quot;A 19 YYYY-MM-DD HH24:MI:SS.&quot;</td>
</tr>
<tr>
<td>Thousand separator</td>
<td>Thousand separator for numeric values. Default is None.</td>
</tr>
<tr>
<td>Decimal separator</td>
<td>Decimal separator for numeric values. Default is a period (.).</td>
</tr>
<tr>
<td>Tracing level</td>
<td>Controls the amount of detail in the mapping log file. Note: This field appears when you edit a flat file source or target in a mapping.</td>
</tr>
</tbody>
</table>

Creating a Flat File Data Object

Create a flat file data object to define the data object columns and rows.

1. Select a project or folder in the Object Explorer view.
2. Click File > New > Data Object.
3. Select Physical Data Objects > Flat File Data Object and click Next.
The New Flat File Data Object dialog box appears.

4. Select Create as Empty.
5. Enter a name for the data object.
6. Optionally, click Browse to select a project or folder for the data object.
7. Click Next.
8. Select a code page that matches the code page of the data in the file.
9. Select Delimited or Fixed-width.
10. If you selected Fixed-width, click Finish. If you selected Delimited, click Next.
11. Configure the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delimiters</td>
<td>Character used to separate columns of data. Use the Other field to enter a different delimiter. Delimiters must be printable characters and must be different from the configured escape character and the quote character. You cannot select unprintable multibyte characters as delimiters.</td>
</tr>
<tr>
<td>Text Qualifier</td>
<td>Quote character that defines the boundaries of text strings. If you select a quote character, the Developer tool ignores delimiters within a pair of quotes.</td>
</tr>
</tbody>
</table>

12. Click Finish.

The data object appears under Data Object in the project or folder in the Object Explorer view.

Importing a Fixed-Width Flat File Data Object

Import a fixed-width flat file data object when you have a fixed-width flat file that defines the metadata you want to include in a mapping, mapplet, or profile.

1. Click File > New > Data Object.
   The New dialog box appears.
2. Select Physical Data Objects > Flat File Data Object and click Next.
   The New Flat File Data Object dialog box appears.
3. Enter a name for the data object.
4. Click Browse and navigate to the directory that contains the file.
5. Click Open.
   The wizard names the data object the same name as the file you selected.
6. Optionally, edit the data object name.
7. Click Next.
8. Select a code page that matches the code page of the data in the file.
10. Optionally, edit the maximum number of rows to preview.
11. Click Next.
12. Configure the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import Field Names From First Line</td>
<td>If selected, the Developer tool uses data in the first row for column names. Select this option if column names appear in the first row.</td>
</tr>
<tr>
<td>Start Import At Row</td>
<td>Row number at which the Data Integration Service starts reading when it imports the file. For example, if you specify to start at the second row, the Developer tool skips the first row before reading.</td>
</tr>
</tbody>
</table>

13. Click **Edit Breaks** to edit column breaks. Or, follow the directions in the wizard to manipulate the column breaks in the file preview window.

You can move column breaks by dragging them. Or, double-click a column break to delete it.

14. Click **Next** to preview the physical data object.

15. Click **Finish**.

The data object appears under Data Object in the project or folder in the **Object Explorer** view.

### Importing a Delimited Flat File Data Object

Import a delimited flat file data object when you have a delimited flat file that defines the metadata you want to include in a mapping, mapplet, or profile.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File > New > Data Object**.

The **New** dialog box appears.

3. Select **Physical Data Objects > Flat File Data Object** and click **Next**.

The **New Flat File Data Object** dialog box appears.

4. Enter a name for the data object.
5. Click **Browse** and navigate to the directory that contains the file.
6. Click **Open**.

The wizard names the data object the same name as the file you selected.

7. Optionally, edit the data object name.
8. Click **Next**.
9. Select a code page that matches the code page of the data in the file.
10. Select Delimited.
11. Optionally, edit the maximum number of rows to preview.
12. Click **Next**.
13. Configure the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delimiters</td>
<td>Character used to separate columns of data. Use the Other field to enter a different delimiter. Delimiters must be printable characters and must be</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>different from the configure escape character and the quote character. You cannot select nonprinting multibyte characters as delimiters.</td>
</tr>
<tr>
<td>Text Qualifier</td>
<td>Quote character that defines the boundaries of text strings. If you select a quote character, the Developer tool ignores delimiters within pairs of quotes.</td>
</tr>
<tr>
<td>Import Field Names From First Line</td>
<td>If selected, the Developer tool uses data in the first row for column names. Select this option if column names appear in the first row. The Developer tool prefixes “FIELD_” to field names that are not valid.</td>
</tr>
<tr>
<td>Row Delimiter</td>
<td>Specify a line break character. Select from the list or enter a character. Precede an octal code with a backslash (). To use a single character, enter the character. The Data Integration Service uses only the first character when the entry is not preceded by a backslash. The character must be a single-byte character, and no other character in the code page can contain that byte. Default is line-feed, ‘\012 LF’ (\n).</td>
</tr>
<tr>
<td>Escape Character</td>
<td>Character immediately preceding a column delimiter character embedded in an unquoted string, or immediately preceding the quote character in a quoted string. When you specify an escape character, the Data Integration Service reads the delimiter character as a regular character.</td>
</tr>
<tr>
<td>Start Import At Row</td>
<td>Row number at which the Data Integration Service starts reading when it imports the file. For example, if you specify to start at the second row, the Developer tool skips the first row before reading.</td>
</tr>
<tr>
<td>Treat Consecutive Delimiters as One</td>
<td>If selected, the Data Integration Service reads one or more consecutive column delimiters as one. Otherwise, the Data Integration Service reads two consecutive delimiters as a null value.</td>
</tr>
<tr>
<td>Remove Escape Character From Data</td>
<td>Removes the escape character in the output string.</td>
</tr>
</tbody>
</table>

14. Click **Next** to preview the data object.
15. Click **Finish**.

The data object appears under Data Object in the project or folder in the **Object Explorer** view.

### WSDL Data Object

A WSDL data object is a physical data object that uses a WSDL file as a source. You can use a WSDL data object to create a Web Service Consumer transformation. Import a WSDL file to create a WSDL data object.

After you import a WSDL data object, you can edit general and advanced properties in the **Overview** and **Advanced** views. The **WSDL** view displays the WSDL file content.
The following figure shows a sample WSDL data object:

Consider the following guidelines when you import a WSDL:

- The WSDL file must be WSDL 1.1 compliant.
- The WSDL file must be valid.
- Operations that you want to include in a Web Service Consumer transformation must use Document/Literal encoding. The WSDL import fails if all operations in the WSDL file use an encoding type other than Document/Literal.
- The Developer tool must be able to access any schema that the WSDL file references.
- If a WSDL file contains a schema or has an external schema, the Developer tool creates an embedded schema within the WSDL data object.
- If a WSDL file imports another WSDL file, the Developer tool combines both WSDLs to create the WSDL data object.
- If a WSDL file defines multiple operations, the Developer tool includes all operations in the WSDL data object.

**WSDL Data Object Overview View**

The WSDL data object **Overview** view displays general information about the WSDL and operations in the WSDL.

The following table describes the general properties that you configure for a WSDL data object:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the WSDL data object.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the WSDL data object.</td>
</tr>
</tbody>
</table>
The following table describes the columns for operations defined in the WSDL data object:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>The location where the WSDL defines the message format and protocol for the operation.</td>
</tr>
<tr>
<td>Input</td>
<td>The WSDL message name associated with the operation input.</td>
</tr>
<tr>
<td>Output</td>
<td>The WSDL message name associated with the operation output.</td>
</tr>
<tr>
<td>Fault</td>
<td>The WSDL message name associated with the operation fault.</td>
</tr>
</tbody>
</table>

**WSDL Data Object Advanced View**

The WSDL data object **Advanced** view displays advanced properties for a WSDL data object.

The following table describes the advanced properties for a WSDL data object:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Default web service connection for a Web Service Consumer transformation.</td>
</tr>
<tr>
<td>File Location</td>
<td>Location where the WSDL file exists.</td>
</tr>
</tbody>
</table>

**Importing a WSDL Data Object**

To create a Web Service Consumer transformation, import a WSDL data object. You can import a WSDL data object from a WSDL file or a URI that points to the WSDL location. You can import a WSDL data object from a WSDL file that contains either a SOAP 1.1 or SOAP 1.2 binding operation or both.

1. Click **File > New > Data Object**.
2. Select **WSDL data object** and click **Next**. The New WSDL Data Object dialog box appears.
3. Click **Browse** next to the **WSDL** option and enter the location of the WSDL. Then, click **OK**. When you enter the location of the WSDL, you can browse to the WSDL file or you can enter the URI to the WSDL.
   - **Note**: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.
4. Enter a name for the WSDL.
5. Click **Browse** next to the **Location** option to select the project or folder location where you want to import the WSDL data object.
6. Click **Next** to view the operations in the WSDL.
7. Click **Finish**. The data object appears under **Physical Data Object** in the project or folder in the **Object Explorer** view.
WSDL Synchronization

You can synchronize a WSDL data object when the WSDL files change. When you synchronize a WSDL data object, the Developer tool reimports the object metadata from the WSDL files.

You use a WSDL data object to create a Web Service Consumer transformation. When you update a WSDL data object, the Developer tool updates the objects that reference the WSDL and marks them as changed when you open them. When the Developer tool compares the new WSDL with the old WSDL, it identifies WSDL components through the name attributes.

If no name attribute changes, the Developer tool updates the objects that reference the WSDL components. For example, you edit a WSDL file and change the type for simple element "CustID" from xs:string to xs:integer. When you synchronize the WSDL data object, the Developer tool updates the element type in all Web Service Consumer transformations that reference the CustID element.

If a name attribute changes, the Developer tool marks the objects that reference the WSDL component as changed when you open them.

The Developer tool validates the WSDL files before it updates the WSDL data object. If the WSDL files contain errors, the Developer tool does not import the files.

Synchronizing a WSDL Data Object

Synchronize a WSDL data object when the WSDL files change.

1. Right-click the WSDL data object in the Object Explorer view, and select Synchronize.
   The Synchronize WSDL Data Object dialog box appears.
2. Click Browse next to the WSDL field, and enter the location of the WSDL. Then, click OK.
   When you enter the location of the WSDL, you can browse to the WSDL file or you can enter the URI to the WSDL.
   Note: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.
3. Verify the WSDL name and location.
4. Click Next to view the operations in the WSDL.
5. Click Finish.
   The Developer tool updates the objects that reference the WSDL and marks them as changed when you open them.

Certificate Management

The Developer tool must use a certificate to import WSDL data objects and schema objects from a URL that requires client authentication.

By default, the Developer tool imports objects from URLs that require client authentication when the server that hosts the URL uses a trusted certificate. When the server that hosts the URL uses an untrusted certificate, add the untrusted certificate to the Developer tool. If you do not add the untrusted certificate to the Developer tool, the Developer tool cannot import the object. Request the certificate file and password from the server administrator for the URL that you want import objects from.

The certificates that you add to the Developer tool apply to imports that you perform on the Developer tool machine. The Developer tool does not store certificates in the Model repository.
Informatica Developer Certificate Properties

Add certificates to the Developer tool when you want to import objects from a URL that requires client authentication with an untrusted certificate.

The following table describes the certificate properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Name of the server that hosts the URL.</td>
</tr>
<tr>
<td>Port Number</td>
<td>Port number of the URL.</td>
</tr>
<tr>
<td>Certificate File Path</td>
<td>Location of the client certificate file.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the client certificate file.</td>
</tr>
</tbody>
</table>

Adding Certificates to Informatica Developer

When you add a certificate, you configure the certificate properties that the Developer tool uses when you import objects from a URL that requires client authentication with an untrusted certificate.

1. Click Windows > Preferences.
2. Select Informatica > Web Services > Certificates.
3. Click Add.
4. Configure the certificate properties.
5. Click OK.

Synchronization

You can synchronize physical data objects when their sources change. When you synchronize a physical data object, the Developer tool reimports the object metadata from the source you select.

You can synchronize all physical data objects. When you synchronize relational data objects or customized data objects, you can retain or overwrite the key relationships you define in the Developer tool.

You can configure a customized data object to be synchronized when its sources change. For example, a customized data object uses a relational data object as a source, and you add a column to the relational data object. The Developer tool adds the column to the customized data object. To synchronize a customized data object when its sources change, select the Synchronize input and output option in the Overview properties of the customized data object.

To synchronize any physical data object, right-click the object in the Object Explorer view, and select Synchronize.

Synchronizing a Flat File Data Object

You can synchronize the changes to an external flat file data source with its data object in Informatica Developer. Use the Synchronize Flat File wizard to synchronize the data objects.

1. In the Object Explorer view, select a flat file data object.
2. Right-click and select **Synchronize**.
The **Synchronize Flat File Data Object** wizard appears.
3. Verify the flat file path in the **Select existing flat file** field.
4. Click **Next**.
5. Optionally, select the code page, format, delimited format properties, and column properties.
6. Click **Finish**, and then click **OK**.

**Synchronizing a Relational Data Object**

You can synchronize external data source changes of a relational data source with its data object in Informatica Developer. External data source changes include adding, changing, and removing columns, and changes to rules.

1. In the **Object Explorer** view, select a relational data object.
2. Right-click and select **Synchronize**.
   
   A message prompts you to confirm the action.
3. To complete the synchronization process, click **OK**. Click **Cancel** to cancel the process.
   
   If you click **OK**, a synchronization process status message appears.
4. When you see a **Synchronization complete** message, click **OK**.
   
   The message displays a summary of the metadata changes made to the data object.

**Troubleshooting Physical Data Objects**

I am trying to preview a relational data object or a customized data object source transformation and the preview fails.

Verify that the resource owner name is correct.

When you import a relational resource, the Developer tool imports the owner name when the user name and schema from which the table is imported do not match. If the user name and schema from which the table is imported match, but the database default schema has a different name, preview fails because the Data Integration Service executes the preview query against the database default schema, where the table does not exist.

Update the relational data object or the source transformation and enter the correct resource owner name. The owner name appears in the relational data object or the source transformation **Advanced** properties.

I am trying to preview a flat file data object and the preview fails. I get an error saying that the system cannot find the path specified.

Verify that the machine that hosts Informatica services can access the source file directory.

For example, you create a flat file data object by importing the following file on your local machine, MyClient:

```
C:\MySourceFiles\MyFile.csv
```

In the Read view, select the Runtime properties in the Output transformation. The source file directory is "C:\MySourceFiles."

When you preview the file, the Data Integration Service tries to locate the file in the "C:\MySourceFiles" directory on the machine that hosts Informatica services. If the directory does not exist on the machine that hosts Informatica services, the Data Integration Service returns an error when you preview the file.
To work around this issue, use the network path as the source file directory. For example, change the source file directory from "C:\MySourceFiles" to "\MyClient\MySourceFiles." Share the "MySourceFiles" directory so that the machine that hosts Informatica services can access it.
CHAPTER 5

Schema Object

This chapter includes the following topics:
- Schema Object Overview, 78
- Schema Object Overview View, 78
- Schema Object Schema View, 79
- Schema Object Advanced View, 84
- Importing a Schema Object, 85
- Schema Updates, 85
- Certificate Management, 88

Schema Object Overview

A schema object is an XML schema that you import to the Model repository. After you import the schema, you can view the schema components in the Developer tool.

When you import a schema you can edit general schema properties in the Overview view. Edit advanced properties in the Advanced view. View the schema file content in the Schema view.

Schema Object Overview View

Select the Overview view to update the schema name or schema description, view namespaces, and manage schema files.

The Overview view shows the name, description, and target namespace for the schema. You can edit the schema name and the description. The target namespace displays the namespace to which the schema components belong. If no target namespace appears, the schema components do not belong to a namespace.
The following figure shows the Overview view of a schema object:

The Schema Locations area lists the schema files and the namespaces. You can add multiple root .xsd files. If a schema file includes or imports other schema files, the Developer tool includes the child .xsd files in the schema. The namespace associated with each schema file differentiates between elements that come from different sources but have the same names. A Uniform Resource Identifier (URI) reference defines the location of the file that contains the elements and attribute names.

Schema Files

You can add multiple root-level .xsd files to a schema object. You can also remove root-level .xsd files from a schema object.

When you add a schema file, the Developer tool imports all .xsd files that are imported by or included in the file you add. The Developer tool validates the files you add against the files that are part of the schema object. The Developer tool does not allow you to add a file if the file conflicts with a file that is part of the schema object.

For example, a schema object contains root schema file "BostonCust.xsd." You want to add root schema file "LACust.xsd" to the schema object. Both schema files have the same target namespace and define an element called "Customer." When you try to add schema file LACust.xsd to the schema object, the Developer tool prompts you to retain the BostonCust.xsd file or overwrite it with the LACust.xsd file.

You can remove any root-level schema file. If you remove a schema file, the Developer tool changes the element type of elements that were defined by the schema file to xs:string.

To add a schema file, select the Overview view, and click the Add button next to the Schema Locations list. Then, select the schema file. To remove a schema file, select the file and click the Remove button.

Schema Object Schema View

The Schema view shows an alphabetic list of the groups, elements, types, attribute groups, and attributes in the schema. When you select a group, element, type, attribute group, or attribute in the Schema view, properties display in the right panel. You can also view each .xsd file in the Schema view.

The Schema view provides a list of the namespaces and the .xsd files in the schema object.
The following figure shows the **Schema** view of a schema object:

![Schema view](image)

You can perform the following actions in the **Schema** view:

- To view the list of schema constructs, expand the **Directives** folder. To view the namespace, prefix, and the location, select a schema construct from the list.
- To view the namespace prefix, generated prefix, and location, select a namespace. You can change the generated prefix.
- To view the schema object as an .xsd file, select **Source**. If the schema object includes other schemas, you can select which .xsd file to view.
- To view an alphabetic list of groups, elements, types, attribute groups, and attributes in each namespace of the schema, select **Design**. You can enter one or more characters in the **Name** field to filter the groups, elements, types, attribute groups, and attributes by name.
- To view the element properties, select a group, element, type, attribute group, or attribute. The Developer tool displays different fields in the right panel based on the object you select.

When you view types, you can see whether a type is derived from another type. The interface shows the parent type. The interface also shows whether the child element inherited values by restriction or extension.

**Namespace Properties**

The Namespace view shows the prefix and location for a selected namespace.

When you import an XML schema that contains more than one namespace, the Developer tool adds the namespaces to the schema object. When the schema file includes other schemas, the namespaces for those schemas are also included.

The Developer tool creates a generated prefix for each namespace. When the XML schema does not contain a prefix, the Developer tool generates the namespace prefix tns0 and increments the prefix number for each additional namespace prefix. The Developer tool reserves the namespace prefix xs. If you import an XML schema that contains the namespace prefix xs, the Developer tool creates the generated prefix xs1. The Developer tool increments the prefix number when the schema contains the generated prefix value.
For example, Customer_Orders.xsd has a namespace. The schema includes another schema, Customers.xsd. The Customers schema has a different namespace. The Developer tool assigns prefix tns0 to the Customer_Orders namespace and prefix tns1 to the Customers namespace.

To view the namespace location and prefix, select a namespace in the Schema view.

Element Properties

An element is a simple or a complex type. A complex type contains other types. When you select an element in the Schema view, the Developer tool lists the child elements and the properties in the right panel of the screen.

The following table describes element properties that appear when you select an element:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The element name.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the type.</td>
</tr>
<tr>
<td>Type</td>
<td>The element type.</td>
</tr>
</tbody>
</table>

The following table describes the child element properties that appear when you select an element:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The element name.</td>
</tr>
<tr>
<td>Type</td>
<td>The element type.</td>
</tr>
<tr>
<td>Minimum Occurs</td>
<td>The minimum number of times that the element can occur at one point in an XML instance.</td>
</tr>
<tr>
<td>Maximum Occurs</td>
<td>The maximum number of times that the element can occur at one point in an XML instance.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the element.</td>
</tr>
</tbody>
</table>

To view additional child element properties, click the double arrow in the Description column to expand the window.

The following table describes the additional child element properties that appear when you expand the Description column:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Value</td>
<td>A specific value for an element that does not change.</td>
</tr>
<tr>
<td>Nillable</td>
<td>The element can have nil values. A nil element has element tags but has no value and no content.</td>
</tr>
<tr>
<td>Abstract</td>
<td>The element is an abstract type. An XML instance must include types derived from that type. An abstract type is not a valid type without derived element types.</td>
</tr>
<tr>
<td>Minimum Value</td>
<td>The minimum value for an element in an XML instance.</td>
</tr>
<tr>
<td>Maximum Value</td>
<td>The maximum value for an element in an XML instance.</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>The minimum length of an element. Length is in bytes, characters, or items based on the element type.</td>
</tr>
</tbody>
</table>
**Property** | **Description**
--- | ---
Maximum Length | The maximum length of an element. Length is in bytes, characters, or items based on the element type.
Enumeration | A list of all legal values for an element.
Pattern | An expression pattern that defines valid element values.

**Element Advanced Properties**

To view advanced properties for a element, select the element in the **Schema** view. Click **Advanced**.

The following table describes the element advanced properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>The element is an abstract type. A SOAP message must include types derived from that type. An abstract type is not a valid type without derived element types.</td>
</tr>
<tr>
<td>Block</td>
<td>Prevents a derived element from appearing in the XML in place of this element. The block value can contain &quot;#all&quot; or a list that includes extension, restriction, or substitution.</td>
</tr>
<tr>
<td>Final</td>
<td>Prevents the schema from extending or restricting the simple type as a derived type.</td>
</tr>
<tr>
<td>Substitution Group</td>
<td>The name of an element to substitute with the element.</td>
</tr>
<tr>
<td>Nillible</td>
<td>The element can have nil values. A nil element has element tags but has no value and no content.</td>
</tr>
</tbody>
</table>

**Simple Type Properties**

A simple type element is an XML element that contains unstructured text. When you select a simple type element in the **Schema** view, information about the simple type element appears in the right panel.

The following table describes the properties you can view for a simple type:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name of the element.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the element.</td>
</tr>
<tr>
<td>Variety</td>
<td>Defines if the simple type is union, list, anyType, or atomic. An atomic element contains no other elements or attributes.</td>
</tr>
<tr>
<td>Member types</td>
<td>A list of the types in a UNION construct.</td>
</tr>
<tr>
<td>Item type</td>
<td>The element type.</td>
</tr>
<tr>
<td>Base</td>
<td>The base type of an atomic element, such as integer or string.</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>The minimum length for an element. Length is in bytes, characters, or items based on the element type.</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>The maximum length for an element. Length is in bytes, characters, or items based on the element type.</td>
</tr>
</tbody>
</table>
### Simple Type Advanced Properties

To view advanced properties for a simple type, select the simple type in the Schema view. Click **Advanced**.

The advanced properties appear below the simple type properties.

The following table describes the advanced property for a simple type:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final</td>
<td>Prevents the schema from extending or restricting the simple type as a derived type.</td>
</tr>
</tbody>
</table>

### Complex Type Properties

A complex type is an XML element that contains other elements and attributes. A complex type contains elements that are simple or complex types. When you select a complex type in the Schema view, the Developer tool lists the child elements and the child element properties in the right panel of the screen.

The following table describes complex type properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The type name.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the type.</td>
</tr>
<tr>
<td>Inherit from</td>
<td>Name of the parent type.</td>
</tr>
<tr>
<td>Inherit by</td>
<td>Restriction or extension. A complex type is derived from a parent type. The complex type might reduce the elements or attributes of the parent. Or, it might add elements and attributes.</td>
</tr>
</tbody>
</table>

To view properties of each element in a complex type, click the double arrow in the Description column to expand the window.

### Complex Type Advanced Properties

To view advanced properties for a complex type, select the element in the Schema view. Click **Advanced**.

The following table describes the advanced properties for a complex element or type:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>The element is an abstract type. A SOAP message must include types derived from that type. An abstract type is not a valid type without derived element types.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Block</td>
<td>Prevents a derived element from appearing in the XML in place of this element. The block value can contain &quot;#all&quot; or a list that includes extension, restriction, or substitution.</td>
</tr>
<tr>
<td>Final</td>
<td>Prevents the schema from extending or restricting the simple type as a derived type.</td>
</tr>
<tr>
<td>Substitution Group</td>
<td>The name of an element to substitute with the element.</td>
</tr>
<tr>
<td>Nilible</td>
<td>The element can have nil values. A nil element has element tags but has no value and no content.</td>
</tr>
</tbody>
</table>

### Attribute Properties

An attribute is a simple type. Elements and complex types contain attributes. Global attributes appear as part of the schema. When you select a global attribute in the Schema view, the Developer tool lists attribute properties and related type properties in the right panel of the screen.

The following table describes the attribute properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The attribute name.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the attribute.</td>
</tr>
<tr>
<td>Type</td>
<td>The attribute type.</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the attribute type. Indicates whether the value of the attribute type is fixed or has a default value. If no value is defined, the property displays default=0.</td>
</tr>
</tbody>
</table>

The following table describes the type properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Length</td>
<td>The minimum length of the type. Length is in bytes, characters, or items based on the type.</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>The maximum length of the type. Length is in bytes, characters, or items based on the type.</td>
</tr>
<tr>
<td>Collapse Whitespace</td>
<td>Strips leading and trailing whitespace. Collapses multiple spaces to a single space.</td>
</tr>
<tr>
<td>Enumerations</td>
<td>Restrict the type to the list of legal values.</td>
</tr>
<tr>
<td>Patterns</td>
<td>Restrict the type to values defined by a pattern expression.</td>
</tr>
</tbody>
</table>

### Schema Object Advanced View

View advanced properties for the schema object.
The following table describes advanced properties for a schema object:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elementFormDefault</td>
<td>Qualified or Unqualified</td>
<td>Determines whether or not elements must have a namespace. The schema qualifies elements with a prefix or by a target namespace declaration. The unqualified value means that the elements do not need a namespace.</td>
</tr>
<tr>
<td>attributeFormDefault</td>
<td>Qualified or Unqualified</td>
<td>Determines whether or not locally declared attributes must have a namespace. The schema qualifies attributes with a prefix or by a target namespace declaration. The unqualified value means that the attributes do not need a namespace.</td>
</tr>
<tr>
<td>File location</td>
<td>Full path to the .xsd file</td>
<td>The location of the .xsd file when you imported it.</td>
</tr>
</tbody>
</table>

**Importing a Schema Object**

You can import an .xsd file to create a schema object in the repository.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File > New > Schema**.
   The **New Schema** dialog box appears.
3. Browse and select an .xsd file to import.
   You can enter a URI or a location on the file system to browse. The Developer tool validates the schema you choose. Review validation messages.
   **Note**: If the URI contains non-English characters, the import might fail. Copy the URI to the address bar in any browser. Copy the location back from the browser. The Developer tool accepts the encoded URI from the browser.
4. Click **OK**.
   The schema name appears in the dialog box.
5. Optionally, change the schema name.
6. Click **Next** to view a list of the elements and types in the schema.
7. Click **Finish** to import the schema.
   The schema appears under Schema Objects in the **Object Explorer** view.
8. To change the generated prefix for a schema namespace, select the namespace in the **Object Explorer** view.
   Change the **Generated Prefix** property in the **Namespace** view.

**Schema Updates**

You can update a schema object when elements, attributes, types, or other schema components change. When you update a schema object, the Developer tool updates objects that use the schema.

You can update a schema object through the following methods:
Synchronize the schema.

Synchronize a schema object when you update the schema files outside the Developer tool. When you synchronize a schema object, the Developer tool reimports all of the schema .xsd files that contain changes.

Edit a schema file.

Edit a schema file when you want to update a file from within the Developer tool. When you edit a schema file, the Developer tool opens the file in the editor you use for .xsd files. You can open the file in a different editor or set a default editor for .xsd files in the Developer tool.

Schema Synchronization

You can synchronize a schema object when the schema components change. When you synchronize a schema object, the Developer tool reimports the object metadata from the schema files.

Use schema synchronization when you make complex changes to the schema object outside the Developer tool. For example, you might synchronize a schema after you perform the following actions:

- Make changes to multiple schema files.
- Add or remove schema files from the schema.
- Change import or include elements.

The Developer tool validates the schema files before it updates the schema object. If the schema files contain errors, the Developer tool does not import the files.

To synchronize a schema object, right-click the schema object in the Object Explorer view, and select Synchronize.

Schema File Edits

You can edit a schema file from within the Developer tool to update schema components.

Edit a schema file in the Developer tool to make minor updates to a small number of files. For example, you might make one of the following minor updates to a schema file:

- Change the minOccurs or maxOccurs attributes for an element.
- Add an attribute to a complex type.
- Change a simple object type.

When you edit a schema file, the Developer tool opens a temporary copy of the schema file in an editor. You can edit schema files with the system editor that you use for .xsd files, or you can select another editor. You can also set the Developer tool default editor for .xsd files. Save the temporary schema file after you edit it.

The Developer tool validates the temporary file before it updates the schema object. If the schema file contains errors or contains components that conflict with other schema files in the schema object, the Developer tool does not import the file.

**Note:** When you edit and save the temporary schema file, the Developer tool does not update the schema file that appears in the Schema Locations list. If you synchronize a schema object after you edit a schema file in the Developer tool, the synchronization operation overwrites your edits.

Setting a Default Schema File Editor

You can set the default editor that the Developer tool opens when you edit a schema file.

1. Click Window > Preferences.
   
   The Preferences dialog box appears.
2. Click Editors > File Associations.

   The File Associations page of the Preferences dialog box appears.

3. Click Add next to the File types area.

   The Add File Type dialog box appears.

4. Enter .xsd as the file type, and click OK.

5. Click Add next to the Associated editors area.

   The Editor Selection dialog box appears.

6. Select an editor from the list of editors or click Browse to select a different editor, and then click OK.

   The editor that you select appears in the Associated editors list.

7. Optionally, add other editors to the Associated editors list.

8. If you add multiple editors, you can change the default editor. Select an editor, and click Default.

9. Click OK.

**Editing a Schema File**

You can edit any schema file in a schema object.

1. Open a schema object.

2. Select the Overview view.
3. Select a schema file in the **Schema Locations** list.

4. Click **Open with**, and select one of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Editor</td>
<td>The schema file opens in the editor that your operating system uses for .xsd files.</td>
</tr>
<tr>
<td>Default Editor</td>
<td>The schema file opens in the editor that you set as the default editor in the Developer tool. This option appears if you set a default editor.</td>
</tr>
<tr>
<td>Other</td>
<td>You select the editor in which to open the schema file.</td>
</tr>
</tbody>
</table>

The Developer tool opens a temporary copy of the schema file.

5. Update the temporary schema file, save the changes, and close the editor.

The Developer tool prompts you to update the schema object.

6. To update the schema object, click **Update Schema Object**.

The Developer tool updates the schema file with the changes you made.

---

**Certificate Management**

The Developer tool must use a certificate to import WSDL data objects and schema objects from a URL that requires client authentication.

By default, the Developer tool imports objects from URLs that require client authentication when the server that hosts the URL uses a trusted certificate. When the server that hosts the URL uses an untrusted certificate, add the untrusted certificate to the Developer tool. If you do not add the untrusted certificate to the Developer tool, the Developer tool cannot import the object. Request the certificate file and password from the server administrator for the URL that you want to import objects from.

The certificates that you add to the Developer tool apply to imports that you perform on the Developer tool machine. The Developer tool does not store certificates in the Model repository.
Informatica Developer Certificate Properties

Add certificates to the Developer tool when you want to import objects from a URL that requires client authentication with an untrusted certificate.

The following table describes the certificate properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Name of the server that hosts the URL.</td>
</tr>
<tr>
<td>Port Number</td>
<td>Port number of the URL.</td>
</tr>
<tr>
<td>Certificate File Path</td>
<td>Location of the client certificate file.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the client certificate file.</td>
</tr>
</tbody>
</table>

Adding Certificates to Informatica Developer

When you add a certificate, you configure the certificate properties that the Developer tool uses when you import objects from a URL that requires client authentication with an untrusted certificate.

1. Click **Windows > Preferences**.
2. Select **Informatica > Web Services > Certificates**.
3. Click **Add**.
4. Configure the certificate properties.
5. Click **OK**.
Profiles Overview

A column profile determines the characteristics of columns in a data source, such as value frequency, percentages, and patterns.

Column profiling discovers the following facts about data:

- The number of unique and null values in each column, expressed as a number and a percentage.
- The patterns of data in each column and the frequencies with which these values occur.
- Statistics about the column values, such as the maximum and minimum lengths of values and the first and last values in each column.

Use column profile options to select the columns on which you want to run a profile, set data sampling options, and set drill-down options when you create a profile.

A rule is business logic that defines conditions applied to source data when you run a profile. You can add a rule to the profile to validate data.

Profile Views

You can view and add information about a profile in Informatica Developer using the Overview, Definition, Comments, and Results views.
The following figure shows the profile views in the editor:

When you open a profile from the **Object Explorer** view, the editor on the right pane shows the profile information under the following views:

**Overview**
View and provide general information about the profile, such as name, description, and location.

**Definition**
View and set up the profile definition. This information includes the list of filters and rules assigned to the profile.

**Results**
Shows the results of profile run. You can export the results after you run a profile.

**Comments**
View and add comments to the profile.

---

**Column Profiling Process**

As part of the column profiling process, you can choose to either include all the source columns for profiling or select specific columns. You can also accept the default profile options or configure the profile results, sampling, and drill-down options.

The following steps describe the column profiling process:

1. Select the data object you want to profile.
2. Determine whether you want to create a profile with default options or change the default profile options.
3. Choose where you want to save the profile.
4. Select the columns you want to profile.
5. Select the profile results option.
6. Choose the sampling options.
7. Choose the drill-down options.
8. Define a filter to determine the rows that the profile reads at run time.
9. Run the profile.

Column Profile Options

When you create a profile with the **Column Profiling** option, you can use the profile wizard to define filter and sampling options. These options determine how the profile reads rows from the data set.

The following figure shows a sample filter definition in a profile:

![Filter Definition Example](image)

After you complete the steps in the profile wizard, you can add a rule to the profile. The rule can have the business logic to perform data transformation operations on the data before column profiling.

The following figure shows a rule titled Rule_FullName that merges the LastName and FirstName columns into the Fullname column:

![Rule Example](image)
Rules

Create and apply rules within profiles. A rule is business logic that defines conditions applied to data when you run a profile. Use rules to further validate the data in a profile and to measure data quality progress.

You can add a rule after you create a profile. Add rules to a profile by selecting a reusable rule. To create a rule, validate a mapplet as a rule.

Creating a Rule in Informatica Developer

You need to validate a mapplet as a rule to create a rule in the Developer tool.

Create a mapplet in the Developer tool.
1. Right-click the mapplet editor.
2. Select Validate As > Rule.

Applying a Rule

You can add a rule to a saved column profile.

1. Browse the Object Explorer view and find the profile you need.
2. Right-click the profile and select Open.
   The profile opens in the editor.
3. Click the Definition tab, and select Rules.
4. Click Add.
   The Apply Rule dialog box opens.
5. Click Browse to find the rule you want to apply.
   Select a rule from a repository project, and click OK.
6. Click the Value column under Input Values to select an input port for the rule.
7. Optionally, click the Value column under Output Values to edit the name of the rule output port.
   The rule appears in the Definition tab.

Guidelines for Rules

You can create reusable rules from mapplets in the Developer tool. You can use these rules in profiles to validate source data.

A rule must meet the following requirements:

- It must contain an Input and Output transformation. You cannot use data sources in a rule.
- It can contain Expression transformations, Lookup transformations, and passive transformations. It cannot contain any other type of transformation. For example, a rule cannot contain a Sorter transformation as it is an active transformation.
- It does not specify cardinality between input groups.

Filtering Options

You can add filters to determine the rows that a column profile uses when performing profiling operations. The profile does not process rows that do not meet the filter criteria.

1. Create or open a column profile.
2. Select the **Filter** view.

3. Click **Add**.

4. Select a filter type and click **Next**.

5. Enter a name for the filter. Optionally, enter a text description of the filter.

6. Select **Set as Active** to apply the filter to the profile. Click **Next**.

7. Define the filter criteria.

8. Click **Finish**.

**Sampling Properties**

Configure the sampling properties to determine the number of rows that the profile reads during a profiling operation.

The following table describes the sampling properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Rows</td>
<td>Reads all rows from the source. Default is enabled.</td>
</tr>
<tr>
<td>First</td>
<td>Reads from the first row up to the row you specify.</td>
</tr>
</tbody>
</table>

**Creating a Profile**

You can create a profile for one or more columns in a data object and store the profile object in the Model repository.

1. In the **Object Explorer** view, select the data object you want to profile.

2. Click **File > New > Profile** to open the profile wizard.

3. Select **Profile** and click **Next**.

4. Enter a name for the profile and verify the project location. If required, browse to a new location.

5. Optionally, enter a text description of the profile.

6. Verify that the name of the data object you selected appears in the **Data Objects** section.

7. Click **Next**.

8. Configure the column profile options such as column selection, filter, sampling, and drilldown options.

9. Click **Finish**.

**Column Profile Results**

Column profile analysis provides information about data quality by highlighting value frequencies, patterns and statistics of data.
Column profiling analysis generates the following profile results:

- Percentage and count statistics for unique and null values
- Inferred datatypes
- The datatype that the data source declares for the data
- The maximum and minimum values
- The date and time of the most recent profile run
- Percentage and count statistics for each unique data element in a column
- Percentage and count statistics for each unique character pattern in a column

The following figure shows the column profile results:

### Column Value Properties

Column value properties show the values in the profiled columns and the frequency with which each value appears in each column. The frequencies are shown as a number, a percentage, and a bar chart.

To view column value properties, select Values from the **Show** list. Double-click a column value to drill-down to the rows that contain the value.

The following table describes the properties for column values:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>List of all values for the column in the profile.</td>
</tr>
<tr>
<td>Frequency</td>
<td>Number of times a value appears in a column.</td>
</tr>
<tr>
<td>Percent</td>
<td>Number of times a value appears in a column, expressed as a percentage of all values in the column.</td>
</tr>
<tr>
<td>Chart</td>
<td>Bar chart for the percentage.</td>
</tr>
</tbody>
</table>

### Column Pattern Properties

Column pattern properties show the patterns of data in the profiled columns and the frequency with which the patterns appear in each column. The patterns are shown as a number, a percentage, and a bar chart.

To view pattern information, select Patterns from the **Show** list. Double-click a pattern to drill-down to the rows that contain the pattern.
The following table describes the properties for column value patterns:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patterns</td>
<td>Pattern for the selected column.</td>
</tr>
<tr>
<td>Frequency</td>
<td>Number of times a pattern appears in a column.</td>
</tr>
<tr>
<td>Percent</td>
<td>Number of times a pattern appears in a column, expressed as a percentage of all values in the column.</td>
</tr>
<tr>
<td>Chart</td>
<td>Bar chart for the percentage.</td>
</tr>
</tbody>
</table>

## Column Statistics Properties

Column statistics properties provide maximum and minimum lengths of values and first and last values.

To view statistical information, select Statistics from the Show list.

The following table describes the column statistics properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Length</td>
<td>Length of the longest value in the column.</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>Length of the shortest value in the column.</td>
</tr>
<tr>
<td>Bottom</td>
<td>Last five values in the column.</td>
</tr>
<tr>
<td>Top</td>
<td>First five values in the column.</td>
</tr>
</tbody>
</table>

**Note:** The profile also displays average and standard deviation statistics for columns of type Integer.

## Exporting Profile Results from Informatica Developer

You can export column values and column pattern data from profile results.

Export column values in Distinct Value Count format. Export pattern values in Domain Inference format.

1. In the **Object Explorer** view, select and open a profile.
2. Optionally, run the profile to update the profile results.
3. Select the **Results** view.
4. Select the column that contains the data for export.
5. Under **Details**, select Values or select Patterns and click the **Export** button.
   
   The **Export data to a file** dialog box opens.

6. Accept or change the file name. The default name is `[Profile_name]_[column_name]_DVC` for column value data and `[Profile_name]_[column_name]_DI` for pattern data.
7. Select the type of data to export. You can select either **Values for the selected column** or **Patterns for the selected column**.
8. Under **Save**, select **Save on Client**.
9. Click **Browse** to select a location and save the file locally in your computer. By default, Informatica Developer writes the file to a location set in the Data Integration Service properties of Informatica Administrator.
10. If you do not want to export field names as the first row, clear the **Export field names as first row** check box.
11. Click **OK**.

### Mapplet and Mapping Profiles

You can define a column profile for an object in a mapplet or mapping. Run a profile on a mapplet or a mapping object when you want to verify the design of the mapping or mapplet without saving the profile results. You can also generate a mapping from a profile.

### Generating a Mapping from a Profile

You can create a mapping object from a profile. Use the mapping object you create to develop a valid mapping. The mapping you create has a data source based on the profiled object and can contain transformations based on profile rule logic. After you create the mapping, add objects to complete it.

1. In the **Object Explorer** view, find the profile on which to create the mapping.
2. Right-click the profile name and select **Generate Mapping**.
   
   The **Generate Mapping** dialog box displays.
3. Enter a mapping name. Optionally, enter a description for the mapping.
4. Confirm the folder location for the mapping.
   
   By default, the Developer tool creates the mapping in the **Mappings** folder in the same project as the profile. Click **Browse** to select a different location for the mapping.
5. Confirm the profile definition that the Developer tool uses to create the mapping. To use another profile, click **Select Profile**.
6. Click **Finish**.

The mapping appears in the **Object Explorer**.

Add objects to the mapping to complete it.

### Running a Profile on a Mapplet or Mapping Object

When you run a profile on a mapplet or mapping object, the profile runs on all data columns and enables drill-down operations on the data. You can run a profile on a mapplet or mapping object with multiple output ports.

The profile traces the source data through the mapping to the output ports of the object you selected. The profile analyzes the data that would appear on those ports if you ran the mapping.

1. Open a mapplet or mapping.
2. Verify that the mapplet or mapping is valid.
3. Right-click a data object or transformation and select **Profile Now**.
   
   If the transformation has multiple output groups, the **Select Output Group** dialog box appears. If the transformation has a single output group, the profile results appear on the **Results** tab of the profile.
4. If the transformation has multiple output groups, select the output groups as necessary.
5. Click **OK**.
   The profile results appears in the **Results** tab of the profile.

**Comparing Profiles for Mapping or Mapplet Objects**

You can create a profile that analyzes two objects in a mapplet or mapping and compares the results of the column profiles for those objects.

Like profiles of single mapping or mapplet objects, profile comparisons run on all data columns.

1. Open a mapplet or mapping.
2. Verify that the mapplet or mapping is valid.
3. Press the **CTRL** key and click two objects in the editor.
4. Right-click one of the objects and select **Compare Profiles**.
5. Optionally, configure the profile comparison to match columns from one object to the other object.
6. Optionally, match columns by clicking a column in one object and dragging it onto a column in the other object.
7. Optionally, choose whether the profile analyzes all columns or matched columns only.
8. Click **OK**.
Logical View of Data Overview

A logical view of data is a representation of data that resides in an enterprise. A logical view of data includes a logical data model, logical data objects, and logical data object mappings.

With a logical view of data, you can achieve the following goals:

- Use common data models across an enterprise so that you do not have to redefine data to meet different business needs. It also means if there is a change in data attributes, you can apply this change one time and use one mapping to make this change to all databases that use this data.
- Find relevant sources of data and present the data in a single view. Data resides in various places in an enterprise, such as relational databases and flat files. You can access all data sources and present the data in one view.
- Expose logical data as relational tables to promote reuse.
The following figure shows a sample of related logical data objects:

Logical Data Object Model Example

Create a logical data object model to describe the representation of logical entities in an enterprise. For example, create a logical data object model to present account data from disparate sources in a single view.

American Bank acquires California Bank. After the acquisition, American Bank has the following goals:
- Present data from both banks in a business intelligence report, such as a report on the top 10 customers.
- Consolidate data from both banks into a central data warehouse.

Traditionally, American Bank would consolidate the data into a central data warehouse in a development environment, verify the data, and move the data warehouse to a production environment. This process might take several months or longer. The bank could then run business intelligence reports on the data warehouse in the production environment.

A developer at American Bank can use the Developer tool to create a model of customer, account, branch, and other data in the enterprise. The developer can link the relational sources of American Bank and California bank to a single view of the customer. The developer can then make the data available for business intelligence reports before creating a central data warehouse.

Developing a Logical View of Data

Develop a logical view of data to represent how an enterprise accesses data and uses data.

Before you develop a logical view of data, you can define the physical data objects that you want to use in a logical data object mapping. You can also profile the physical data sources to analyze data quality.

1. Create or import a logical data model.
2. Optionally, add logical data objects to the logical data object model and define relationships between objects.
3. Create a logical data object mapping to read data from a logical data object or write data to a logical data object. A logical data object mapping can contain transformation logic to transform the data.
4. View the output of the logical data object mapping.
Logical Data Object Models

A logical data object model describes the structure and use of data in an enterprise. The model contains logical data objects and defines relationships between them.

Define a logical data object model to create a unified model of data in an enterprise. The data in an enterprise might reside in multiple disparate source systems such as relational databases and flat files. A logical data object model represents the data from the perspective of the business regardless of the source systems. Create a logical data object model to study data, describe data attributes, and define the relationships among attributes.

For example, customer account data from American Bank resides in an Oracle database, and customer account data from California Banks resides in an IBM DB2 database. You want to create a unified model of customer accounts that defines the relationship between customers and accounts. Create a logical data object model to define the relationship.

You can import a logical data object model from a modeling tool. You can also import a logical data object model from an XSD file that you created in a modeling tool. Or, you can manually create a logical data object model in the Developer tool.

You add a logical data object model to a project or folder and store it in the Model repository.

Creating a Logical Data Object Model

Create a logical data object model to define the structure and use of data in an enterprise. When you create a logical data object model, you can add logical data objects. You associate a physical data object with each logical data object. The Developer tool creates a logical data object read mapping for each logical data object in the model.

1. Select a project or folder in the Object Explorer view.
2. Click File > New > Logical Data Object Model.
   The New dialog box appears.
3. Select Logical Data Object Model and click Next.
   The New Logical Data Object Model dialog box appears.
4. Enter a name for the logical data object model.
5. To create logical data objects, click Next. To create an empty logical data object model, click Finish.
   If you click Next, the Developer tool prompts you to add logical data objects to the model.
6. To create a logical data object, click the New button.
   The Developer tool adds a logical data object to the list.
7. Enter a name in the Name column.
8. Optionally, click the Open button in the Data Object column to associate a physical data object with the logical data object.
   The Select a Data Object dialog box appears.
9. Select a physical data object and click OK.
10. Repeat steps 6 through 10 to add logical data objects.
11. Click Finish.
   The logical data object model opens in the editor.
Importing a Logical Data Object Model from a Modeling Tool

You can import a logical data object model from a modeling tool or an XSD file. Import a logical data object model to use an existing model of the structure and data in an enterprise.

1. Select the project or folder to which you want to import the logical data object model.
2. Click File > New > Logical Data Object Model.
   The New Logical Data Object Model dialog box appears.
3. Select Logical Data Object Model from Data Model.
4. Click Next.
5. In the Model Type field, select the modeling tool from which you want to import the logical data object model.
6. Enter a name for the logical data object model.
7. Click Browse to select the location of the logical data object model.
8. Click Next.
9. Browse to the file that you want to import, select the file, and click Open.
10. Configure the import properties.
11. Click Next.
12. Add logical data objects to the logical data object model.
13. Click Finish.
   The logical data objects appear in the editor.

Logical Data Object Model Properties

When you import a logical data object model from a modeling tool, provide the properties associated with the tool.

CA ERwin Data Modeler Import Properties

Configure the import properties when you import a logical data object model from CA ERwin Data Modeler.

The following table describes the properties to configure when you import a model from CA ERwin Data Modeler:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import UDPs</td>
<td>Specifies how to import user-defined properties.</td>
</tr>
<tr>
<td></td>
<td>Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- As metadata. Import an explicit value as the property value object. Explicit values are not exported.</td>
</tr>
<tr>
<td></td>
<td>- As metadata, migrate default values. Import explicit and implicit values as property value objects.</td>
</tr>
<tr>
<td></td>
<td>- In description, migrate default values. Append the property name and value, even if implicit, to the object description property.</td>
</tr>
<tr>
<td></td>
<td>- Both, migrate default values. Import the UDP value, even if implicit, both as metadata and in the object's description.</td>
</tr>
<tr>
<td></td>
<td>Default is As metadata.</td>
</tr>
<tr>
<td>Import relationship name</td>
<td>Specifies how to import the relationship names from ERwin.</td>
</tr>
</tbody>
</table>
IBM Cognos Business Intelligence Reporting - Framework Manager Import Properties

Configure the import properties when you import a logical data object model from IBM Cognos Business Intelligence Reporting - Framework Manager.

The following table describes the properties to configure when you import a model from IBM Cognos Business Intelligence Reporting - Framework Manager:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder Representation</td>
<td>Specifies how to represent folders from the Framework Manager. Select one of the following options: - Ignore. Ignore folders. - Flat. Represent folders as diagrams but do not preserve hierarchy. - Hierarchical. Represent folders as diagrams and preserve hierarchy. Default is Ignore.</td>
</tr>
<tr>
<td>Package Representation</td>
<td>Specifies how to represent packages from Cognos Framework Manager. Select one of the following options: - Ignore. Ignore subject areas. - Subject Areas. Represent packages as subject areas. - Model. Represent the package as the model. Default is Ignore.</td>
</tr>
<tr>
<td>Reverse engineer relationships</td>
<td>Specifies whether the Developer tool computes the relationship between two dbQueries as referential integrity constraints.</td>
</tr>
</tbody>
</table>
### SAP BusinessObjects Designer Import Properties

Configure the import properties when you import a logical data object model from SAP BusinessObjects Designer.

The following table describes the properties to configure when you import a model from SAP BusinessObjects Designer:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Name of the BusinessObjects repository.</td>
</tr>
<tr>
<td></td>
<td>For BusinessObjects versions 11.x and 12.x (XI), enter the name of the Central Management Server. For BusinessObjects version 5.x and 6.x, enter name of the repository defined by the Supervisor application.</td>
</tr>
<tr>
<td>Authentication mode</td>
<td>Login authentication mode. This parameter is applicable to SAP BusinessObjects Designer 11.0 and later. Select one of the following authentication modes:</td>
</tr>
<tr>
<td></td>
<td>- Enterprise. Business Objects Enterprise login</td>
</tr>
<tr>
<td></td>
<td>- LDAP. LDAP server authentication</td>
</tr>
<tr>
<td></td>
<td>- Windows AD. Windows Active Directory server authentication</td>
</tr>
<tr>
<td></td>
<td>- Windows NT. Windows NT domain server authentication</td>
</tr>
<tr>
<td></td>
<td>- Standalone. Standalone authentication</td>
</tr>
<tr>
<td></td>
<td>Default is Enterprise.</td>
</tr>
<tr>
<td>User name</td>
<td>User name in the BusinessObjects server. For version 11.x and 12.x (XI), you need to be a member of BusinessObjects groups.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the BusinessObjects server.</td>
</tr>
<tr>
<td>Silent execution</td>
<td>Specifies whether to execute in interactive or silent mode. Default is Silent.</td>
</tr>
<tr>
<td>Close after execution</td>
<td>Specify whether to close BusinessObjects after the Developer Tool completes the model import.</td>
</tr>
<tr>
<td>Table design level</td>
<td>Specifies the design level of the imported tables. Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- Logical and physical. The tables appear both in the logical view and in the physical view of the model.</td>
</tr>
<tr>
<td></td>
<td>- Physical. The tables appear both in the physical view of the model.</td>
</tr>
<tr>
<td></td>
<td>Default is Physical.</td>
</tr>
<tr>
<td>Transform Joins to Foreign Keys</td>
<td>Transforms simple SQL joins in the model into foreign key relationships.</td>
</tr>
</tbody>
</table>
## Sybase PowerDesigner CDM Import Properties

Configure the import properties when you import a logical data object model from Sybase PowerDesigner CDM.

The following table describes the properties to configure when you import a model from Sybase PowerDesigner CDM:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import UDPs</td>
<td>Specifies how to import user-defined properties. Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- As metadata. Import an explicit value as the property value object. Explicit values are not exported.</td>
</tr>
<tr>
<td></td>
<td>- As metadata, migrate default values. Import explicit and implicit values as property value objects.</td>
</tr>
<tr>
<td></td>
<td>- In description, migrate default values. Append the property name and value, even if implicit, to the object description property.</td>
</tr>
<tr>
<td></td>
<td>- Both, migrate default values. Import the UDP value, even if implicit, both as metadata and in the object's description.</td>
</tr>
<tr>
<td>Default is As metadata.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Import Association Classes</th>
<th>Specifies whether the Developer tool should import association classes.</th>
</tr>
</thead>
</table>

| Import IDs                 | Specifies whether to set the unique ID of the object as the NativeId property.                                                      |
Sybase PowerDesigner OOM 9.x to 15.x Import Properties

Configure the import properties when you import a logical data object model from Sybase PowerDesigner OOM 9.x to 15.x.

When you import a logical data object model from Sybase PowerDesigner OOM, the Developer tool imports the classes and attributes but leaves out other entities. To import a logical data object model, export the model from Sybase PowerDesigner in the UML 1.3 - XMI 1.0 XML format.

The following table describes the properties to configure when you import a model from Sybase PowerDesigner OOM:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Tool</td>
<td>Specifies which tool generated the model you want to import.</td>
</tr>
<tr>
<td></td>
<td>Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- Auto Detect. The Developer tool detects which tool generated the file.</td>
</tr>
<tr>
<td></td>
<td>- OMG XMI. The file conforms to the OMG XMI 1.0 standard DTDs.</td>
</tr>
<tr>
<td></td>
<td>- Argo/UML 0.7. The file was generated by Argo/UML 0.7.0 or earlier.</td>
</tr>
<tr>
<td></td>
<td>- Argo/UML 0.8. The file was generated by Argo/UML 0.7.1 or later.</td>
</tr>
<tr>
<td></td>
<td>- XMI Toolkit. The file was generated by IBM XMI Toolkit.</td>
</tr>
<tr>
<td></td>
<td>- XMI Interchange. The file was generated by Unisys Rose XMI Interchange.</td>
</tr>
<tr>
<td></td>
<td>- Rose UML. The file was generated by Unisys Rose UML.</td>
</tr>
<tr>
<td></td>
<td>- Visio UML. The file was generated by Microsoft Visio Professional 2002 and</td>
</tr>
<tr>
<td></td>
<td>Visio for Enterprise Architects using UML to XMI Export.</td>
</tr>
<tr>
<td></td>
<td>- PowerDesigner UML. The file was generated by Sybase PowerDesigner using XMI</td>
</tr>
<tr>
<td></td>
<td>Export.</td>
</tr>
<tr>
<td></td>
<td>- Component Modeler. The file was generated by CA AllFusion Component Modeler</td>
</tr>
<tr>
<td></td>
<td>using XMI Export.</td>
</tr>
<tr>
<td></td>
<td>- Netbeans XMI Writer. The file was generated by one of applications using</td>
</tr>
<tr>
<td></td>
<td>Netbeans XMI Writer such as Poseidon.</td>
</tr>
<tr>
<td></td>
<td>- Embarcadero Describe. The file was generated by Embarcadero Describe.</td>
</tr>
<tr>
<td></td>
<td>Default is Auto Detect.</td>
</tr>
<tr>
<td>Auto Correct</td>
<td>Fix and import an incomplete or incorrect model in the XML file.</td>
</tr>
<tr>
<td>Model Filter</td>
<td>Model to import if the XML file contains more than one model. Use a comma to</td>
</tr>
<tr>
<td></td>
<td>separate multiple models.</td>
</tr>
<tr>
<td>Top Package</td>
<td>The top-level package in the model.</td>
</tr>
<tr>
<td>Import UUIDs</td>
<td>Import UUIDs as Nativeld.</td>
</tr>
</tbody>
</table>
### Sybase PowerDesigner PDM Import Properties

Configure the import properties when you import a logical data object model from Sybase PowerDesigner PDM.

The following table describes the properties to configure when you import a model from Sybase PowerDesigner PDM:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Import UDPs                  | Specifies how to import user-defined properties. Select one of the following options:  
- As metadata. Import an explicit value as the property value object. Explicit values are not exported.  
- As metadata, migrate default values. Import explicit and implicit values as property value objects.  
- In description, migrate default values. Append the property name and value, even if implicit, to the object description property.  
- Both, migrate default values. Import the UDP value, even if implicit, both as metadata and in the object's description. Default is As metadata.                                                                 |
| Import IDs                    | Specifies whether to set the unique id of the object as the Nativeld property.                                                                                                                                                                                                                                                                |
| Append volumetric information to the description field | Import and append the number of occurrences information to the description property.                                                                                                                                                                                                                      |
| Remove text formatting       | Specifies whether to remove or keep rich text formatting. Select this option if the model was generated by PowerDesigner 7.0 or 7.5 Clear this option if the model was generated by PowerDesigner 8.0 or greater.                                                                                                                                  |

### XSD Import Properties

You can import logical data object models from an XSD file exported by a modeling tool.

The following table describes the properties to configure when you import a model from an XSD file:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements content name</td>
<td>Attribute to hold the textual content like #PCDATA in the XSD file. Default is As metadata.</td>
</tr>
</tbody>
</table>
| Collapse Level         | Specifies when to collapse a class. The value you select determines whether the Developer tool imports all or some of the elements and attributes in the XSD file. Select one of the following options:  
- None. Every XSD element becomes a class and every XSD attribute becomes an attribute.  
- Empty. Only empty classes collapse into the parent classes.  
- Single Attribute. Only XSD elements with a single attribute and no children collapse into the parent class.  
- No Children. Any XSD element that has no child element collapse into the parent class.  
- All. All collapsible XSD elements collapse into the parent class. Default is All.                                                                 |
<p>| Collapse Star          | Specifies whether the Developer tool should collapse XML elements with an incoming xlink into the parent class.                                                                                                                                               |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Type</td>
<td>Specifies whether the Developer tool should create a class type an element collapses into the parent element.</td>
</tr>
<tr>
<td>Any</td>
<td>Specifies whether to create a class or entity for the ‘xs:any’ pseudo-element.</td>
</tr>
<tr>
<td>Generate IDs</td>
<td>Specifies whether to generate additional attributes to create primary and foreign keys.</td>
</tr>
<tr>
<td></td>
<td>By default, the Developer tool does not generate additional attributes.</td>
</tr>
<tr>
<td>Import substitutionGroup as</td>
<td>Specifies how to represent inheritance.</td>
</tr>
<tr>
<td></td>
<td>Select one of the following options:</td>
</tr>
<tr>
<td></td>
<td>- Generalization. Represents inheritance as generalization.</td>
</tr>
<tr>
<td></td>
<td>- Roll down. Duplicate inherited attributes in the subclass.</td>
</tr>
<tr>
<td></td>
<td>Default is Roll down.</td>
</tr>
<tr>
<td>Include Path</td>
<td>Path to the directory that contains the included schema files, if any.</td>
</tr>
<tr>
<td>UDP namespace</td>
<td>Namespace that contains attributes to be imported as user-defined properties.</td>
</tr>
</tbody>
</table>

**Logical Data Objects**

A logical data object is an object in a logical data object model that describes a logical entity in an enterprise. It has attributes, keys, and it describes relationships between attributes.

You include logical data objects that relate to each other in a data object model. For example, the logical data objects Customer and Account appear in a logical data object model for a national bank. The logical data object model describes the relationship between customers and accounts.

In the model, the logical data object Account includes the attribute Account_Number. Account_Number is a primary key, because it uniquely identifies an account. Account has a relationship with the logical data object Customer, because the Customer data object needs to reference the account for each customer.

You can drag a physical data object into the logical data object model editor to create a logical data object. Or, you can create a logical data object and define the attributes and keys.

**Logical Data Object Properties**

A logical data object contains properties that define the data object and its relationship to other logical data objects in a logical data object model.
The logical data object properties are on the tabs in the Properties view. The following figure shows the tabs of a logical data object:

![Logical Data Object Tabs](image)

The following table describes the tabs of a logical data object:

<table>
<thead>
<tr>
<th>Tab Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Name and description of the logical data object.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Comprise the structure of data in a logical data object.</td>
</tr>
<tr>
<td>Keys</td>
<td>One or more attributes in a logical data object can be primary keys or unique keys.</td>
</tr>
<tr>
<td>Relationships</td>
<td>Associations between logical data objects.</td>
</tr>
<tr>
<td>Access</td>
<td>Type of access for a logical data object and each attribute of the data object.</td>
</tr>
<tr>
<td>Mappings</td>
<td>Logical data object mappings associated with a logical data object.</td>
</tr>
</tbody>
</table>

**Attribute Relationships**

A relationship is an association between primary or foreign key attributes of one or more logical data objects.

You can define the following types of relationship between attributes:

**Identifying**

A relationship between two attributes where an attribute is identified through its association with another attribute.

For example, the relationship between the Branch_ID attribute of the logical data object Branch and the Branch_Location attribute of the logical data object Customer is identifying. This is because a branch ID is unique to a branch location.

**Non-Identifying**

A relationship between two attributes that identifies an attribute independently of the other attribute.

For example, the relationship between the Account_Type attribute of the Account logical data object and the Account_Number attribute of the Customer logical data object is non-identifying. This is because you can identify an account type without having to associate it with an account number.
When you define relationships, the logical data object model indicates an identifying relationship as a solid line between attributes. It indicates a non-identifying relationship as a dotted line between attributes.

Creating a Logical Data Object

You can create a logical data object in a logical data object model to define a logical entity in an enterprise.

1. Click File > New > Logical Data Object.
2. Enter a logical data object name.
3. Select the logical data object model for the logical data object and click Finish.
   The logical data object appears in the logical data object model editor.
4. Select the logical data object and click the Properties tab.
5. On the General tab, optionally edit the logical data object name and description.
6. On the Attributes tab, create attributes and specify their datatype and precision.
7. On the Keys tab, optionally specify primary and unique keys for the data object.
8. On the **Relationships** tab, optionally create relationships between logical data objects.

![Relationships Tab Diagram](image)

9. On the **Access** tab, optionally edit the type of access for the logical data object and each attribute in the data object.
   
   Default is read only.

![Access Tab Diagram](image)

10. On the **Mappings** tab, optionally create a logical data object mapping.

    ![Mappings Tab Diagram](image)
Logical Data Object Mappings

A logical data object mapping is a mapping that links a logical data object to one or more physical data objects. It can include transformation logic.

A logical data object mapping can be of the following types:

- Read
- Write

You can associate each logical data object with one logical data object read mapping or one logical data object write mapping.

Logical Data Object Read Mappings

A logical data object read mapping contains one or more physical data objects as input and one logical data object as output. The mapping can contain transformation logic to transform the data.

It provides a way to access data without accessing the underlying data source. It also provides a way to have a single view of data coming from more than one source.

For example, American Bank has a logical data object model for customer accounts. The logical data object model contains a Customers logical data object.

American Bank wants to view customer data from two relational databases in the Customers logical data object. You can use a logical data object read mapping to perform this task and view the output in the Data Viewer view.

Logical Data Object Write Mappings

A logical data object write mapping contains a logical data object as input. It provides a way to write to targets from a logical data object.

The mapping can contain transformation logic to transform the data. The mapping runs without accessing the underlying data target. It provides a single view of the transformed data without writing to the target.

Creating a Logical Data Object Mapping

You can create a logical data object mapping to link data from a physical data object to a logical data object and transform the data.

1. In the Data Object Explorer view, select the logical data object model that you want to add the mapping to.
2. Click File > New > Other.
3. Select Informatica > Data Objects > Data Object Mapping and click Next.
4. Select the logical data object you want to include in the mapping.
5. Select the mapping type.
6. Optionally, edit the mapping name.
7. Click Finish.
   The editor displays the logical data object as the mapping input or output, based on whether the mapping is a read or write mapping.
8. Drag one or more physical data objects to the mapping as read or write objects, based on whether the mapping is a read or write mapping.
9. Optionally, add transformations to the mapping.
10. Link ports in the mapping.
11. Right-click the mapping editor and click Validate to validate the mapping.
   Validation errors appear on the Validation Log view.
12. Fix validation errors and validate the mapping again.
13. Optionally, click the Data Viewer view and run the mapping.
   Results appear in the Output section.
This chapter includes the following topics:

- Transformations Overview, 114
- Developing a Transformation, 116
- Reusable Transformations, 116
- Expressions in Transformations, 117
- Creating a Transformation, 119

Transformations Overview

A transformation is an object that generates, modifies, or passes data.

Informatica Developer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data.

Transformations in a mapping represent the operations that the Data Integration Service performs on the data. Data passes through transformation ports that you link in a mapping or mapplet.

Transformations can be active or passive. Transformations can be connected to the data flow, or they can be unconnected.

For more information, see the PowerCenter Express Transformation Guide.

Active Transformations

An active transformation changes the number of rows that pass through a transformation. Or, it changes the row type.

For example, the Filter transformation is active, because it removes rows that do not meet the filter condition. The Update Strategy transformation is active, because it flags rows for insert, delete, update, or reject.

You cannot connect multiple active transformations or an active and a passive transformation to the same downstream transformation or transformation input group, because the Data Integration Service might not be able to concatenate the rows passed by active transformations.

For example, one branch in a mapping contains an Update Strategy transformation that flags a row for delete. Another branch contains an Update Strategy transformation that flags a row for insert. If you connect these transformations to a single transformation input group, the Data Integration Service cannot combine the delete and insert operations for the row.
**Passive Transformations**

A passive transformation does not change the number of rows that pass through the transformation, and it maintains the row type.

You can connect multiple transformations to the same downstream transformation or transformation input group if all transformations in the upstream branches are passive. The transformation that originates the branch can be active or passive.

**Unconnected Transformations**

Transformations can be connected to the data flow, or they can be unconnected. An unconnected transformation is not connected to other transformations in the mapping. An unconnected transformation is called within another transformation, and returns a value to that transformation.

**Transformation Descriptions**

The Developer tool contains transformations that perform different data integration functions.

The following table describes each transformation:

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td>Active/Connected</td>
<td>Performs aggregate calculations.</td>
</tr>
<tr>
<td>Expression</td>
<td>Passive/Connected</td>
<td>Calculates a value.</td>
</tr>
<tr>
<td>Filter</td>
<td>Active/Connected</td>
<td>Filters data.</td>
</tr>
<tr>
<td>Input</td>
<td>Passive/Connected</td>
<td>Defines mapplet input rows.</td>
</tr>
<tr>
<td>Java</td>
<td>Active or Passive/Connected</td>
<td>Executes user logic coded in Java. The byte code for the user logic is stored in the repository.</td>
</tr>
<tr>
<td>Joiner</td>
<td>Active/Connected</td>
<td>Joins data from different databases or flat file systems.</td>
</tr>
<tr>
<td>Lookup</td>
<td>Active or Passive/Connected or Unconnected</td>
<td>Look up and return data from a flat file, logical data object, reference table, relational table, view, or synonym.</td>
</tr>
<tr>
<td>Output</td>
<td>Passive/Connected</td>
<td>Defines mapplet output rows.</td>
</tr>
<tr>
<td>Rank</td>
<td>Active/Connected</td>
<td>Limits records to a top or bottom range.</td>
</tr>
<tr>
<td>Router</td>
<td>Active/Connected</td>
<td>Routes data into multiple transformations based on group conditions.</td>
</tr>
<tr>
<td>Sequence Generator</td>
<td>Passive/</td>
<td>Generates a numeric sequence of values.</td>
</tr>
</tbody>
</table>
### Developing a Transformation

When you build a mapping, you add transformations and configure them to handle data according to a business purpose.

Complete the following tasks to develop a transformation and incorporate it into a mapping:

1. Add a nonreusable transformation to a mapping or mapplet. Or, create a reusable transformation that you can add to multiple mappings or mapplets.
2. Configure the transformation. Each type of transformation has a unique set of options that you can configure.
3. If the transformation is reusable, add it to the mapping or mapplet.
4. Link the transformation to other objects in the mapping or mapplet.

   You drag ports from upstream objects to the transformation input ports. You drag output ports from the transformation to ports on downstream objects. Some transformations use predefined ports that you can select.

**Note:** If you create a reusable transformation, you add the input and output ports you need before you link the transformation to other objects. You cannot add ports to the transformation instance on the mapplet or mapping canvas. To update the ports on a reusable transformation, open the transformation object from the repository project and add the ports.

### Reusable Transformations

Reusable transformations are transformations that you can use in multiple mappings or mapplets.

For example, you might create an Expression transformation that calculates value-added tax for sales in Canada to analyze the cost of doing business in that country. Rather than perform the same work every time, you can create a

---

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Sorter</td>
<td>Active/</td>
<td>Sorts data based on a sort key.</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>Active or Passive/</td>
<td>Executes SQL queries against a database.</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Union</td>
<td>Active/</td>
<td>Merges data from different databases or flat file systems.</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Update Strategy</td>
<td>Active/</td>
<td>Determines whether to insert, delete, update, or reject rows.</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Web Service Consumer</td>
<td>Active/</td>
<td>Connects to a web service as a web service client to access or transform data.</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td></td>
</tr>
</tbody>
</table>
Reusable transformation. When you need to incorporate this transformation into a mapping, you add an instance of it to the mapping. If you change the definition of the transformation, all instances of it inherit the changes.

The Developer tool stores each reusable transformation as metadata separate from any mapping or mapplet that uses the transformation. It stores reusable transformations in a project or folder.

When you add instances of a reusable transformation to mappings, changes you make to the transformation might invalidate the mapping or generate unexpected data.

Reusable Transformation Instances and Inherited Changes

When you add a reusable transformation to a mapping or mapplet, you add an instance of the transformation. The definition of the transformation still exists outside the mapping or mapplet, while an instance of the transformation appears within the mapping or mapplet.

When you change the transformation, instances of the transformation reflect these changes. Instead of updating the same transformation in every mapping that uses it, you can update the reusable transformation one time, and all instances of the transformation inherit the change. Instances inherit changes to ports, expressions, properties, and the name of the transformation.

Editing a Reusable Transformation

When you edit a reusable transformation, all instances of that transformation inherit the changes. Some changes might invalidate the mappings that use the reusable transformation.

You can open the transformation in the editor to edit a reusable transformation. You cannot edit an instance of the transformation in a mapping. However, you can edit the transformation runtime properties.

If you make any of the following changes to a reusable transformation, mappings that use instances of it might not be valid:

- When you delete one or more ports in a transformation, you disconnect the instance from part or all of the data flow through the mapping.
- When you change a port datatype, you make it impossible to map data from that port to another port that uses an incompatible datatype.
- When you change a port name, expressions that refer to the port are no longer valid.
- When you enter an expression that is not valid in the reusable transformation, mappings that use the transformation are no longer valid. The Data Integration Service cannot run mappings that are not valid.

Expressions in Transformations

You can enter expressions in the Expression Editor in some transformations. Expressions modify data or test whether data matches conditions.

Create expressions that use transformation language functions. Transformation language functions are SQL-like functions that transform data.

Enter an expression in a port that uses the value of data from an input or input/output port. For example, you have a transformation with an input port IN_SALARY that contains the salaries of all the employees. You might use the values from the IN_SALARY column later in the mapping. You might also use the transformation to calculate the total and average salaries. The Developer tool requires you to create a separate output port for each calculated value.
The following table lists the transformations in which you can enter expressions:

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td>Performs an aggregate calculation based on all data passed through the transformation. Alternatively, you can specify a filter for records in the aggregate calculation to exclude certain kinds of records. For example, you can find the total number and average salary of all employees in a branch office using this transformation.</td>
</tr>
<tr>
<td>Expression</td>
<td>Performs a calculation based on values within a single row. For example, based on the price and quantity of a particular item, you can calculate the total purchase price for that line item in an order.</td>
</tr>
<tr>
<td>Filter</td>
<td>Specifies a condition used to filter rows passed through this transformation. For example, if you want to write customer data to the BAD_DEBT table for customers with outstanding balances, you could use the Filter transformation to filter customer data.</td>
</tr>
<tr>
<td>Joiner</td>
<td>Specifies an advanced condition used to join unsorted source data. For example, you can concatenate first name and last name master ports and then match them with the full name detail port.</td>
</tr>
<tr>
<td>Rank</td>
<td>Sets the conditions for rows included in a rank. For example, you can rank the top 10 salespeople who are employed with the organization.</td>
</tr>
<tr>
<td>Router</td>
<td>Routes data into multiple transformations based on a group expression. For example, use this transformation to compare the salaries of employees at three different pay levels. You can do this by creating three groups in the Router transformation. For example, create one group expression for each salary range.</td>
</tr>
<tr>
<td>Update Strategy</td>
<td>Flags a row for update, insert, delete, or reject. You use this transformation when you want to control updates to a target, based on some condition you apply. For example, you might use the Update Strategy transformation to flag all customer rows for update when the mailing address has changed. Or, you might flag all employee rows for reject for people who no longer work for the organization.</td>
</tr>
</tbody>
</table>

The Expression Editor

Use the Expression Editor to build SQL-like statements.

You can enter an expression manually or use the point-and-click method. Select functions, ports, variables, and operators from the point-and-click interface to minimize errors when you build expressions. The maximum number of characters you can include in an expression is 32,767.
Port Names in an Expression

You can enter transformation port names in an expression.

For connected transformations, if you use port names in an expression, the Developer tool updates that expression when you change port names in the transformation. For example, you write an expression that determines the difference between two dates, Date_Promised and Date_Delivered. If you change the Date_Promised port name to Due_Date, the Developer tool changes the Date_Promised port name to Due_Date in the expression.

**Note:** You can propagate the name Due_Date to other non-reusable transformations that depend on this port in the mapping.

Adding an Expression to a Port

You can add an expression to an output port.

1. In the transformation, select the port and open the **Expression Editor**.
2. Enter the expression.
   - Use the Functions and Ports tabs and the operator keys.
3. Optionally, add comments to the expression.
   - Use comment indicators -- or //.
4. Click the Validate button to validate the expression.
5. Click **OK**.
6. If the expression is not valid, fix the validation errors and validate the expression again.
7. When the expression is valid, click **OK** to close the **Expression Editor**.

Comments in an Expression

You can add comments to an expression to describe the expression or to specify a valid URL to access business documentation about the expression.

To add comments within the expression, use -- or // comment indicators.

Expression Validation

You need to validate an expression to run a mapping or preview mapplet output.

Use the Validate button in the **Expression Editor** to validate an expression. If you do not validate an expression, the Developer tool validates it when you close the **Expression Editor**. If the expression is invalid, the Developer tool displays a warning. You can save the invalid expression or modify it.

Creating a Transformation

You can create a reusable transformation to reuse in multiple mappings or mapplets. Or, you can create a non-reusable transformation to use one time in a mapping or mapplet.

To create a reusable transformation, click **File > New > Transformation** and complete the wizard.

To create a non-reusable transformation in a mapping or mapplet, select a transformation from the Transformation palette and drag the transformation to the editor.
Certain transformations require you to choose a mode or perform additional configuration when you create the transformation. For example, the Parser transformation requires that you choose either token parsing mode or pattern parsing mode when you create the transformation.

After you create a transformation, it appears in the editor. Some transformations contain predefined ports and groups. Other transformations are empty.
Mappings Overview

A mapping is a set of inputs and outputs that represent the data flow between sources and targets. They can be linked by transformation objects that define the rules for data transformation. The Data Integration Service uses the instructions configured in the mapping to read, transform, and write data.

The type of input and output you include in a mapping determines the type of mapping. You can create the following types of mappings in the Developer tool:

- Mapping with physical data objects as the input and output
- Logical data object mapping with a logical data object as the mapping input or output

Note: You can include a mapping with physical data objects as the input and output in a Mapping task in a workflow. You might want to run a mapping from a workflow so that you can run multiple mappings sequentially. Or, you can develop a workflow that runs commands to perform steps before and after a mapping runs.
The following figure shows an example of a mapping:

![Mapping Example](image)

**Object Dependency in a Mapping**

A mapping is dependent on some objects that are stored as independent objects in the repository.

When object metadata changes, the Developer tool tracks the effects of these changes on mappings. Mappings might become invalid even though you do not edit the mapping. When a mapping becomes invalid, the Data Integration Service cannot run it.

The following objects are stored as independent objects in the repository:

- Logical data objects
- Physical data objects
- Reusable transformations
- Mapplets

A mapping is dependent on these objects.

Non-reusable transformations that you build within the mapping are stored as dependent repository objects within the mapping.

**Developing a Mapping**

Develop a mapping to read, transform, and write data according to your business needs.

1. Determine the type of mapping that you want to create.
2. Create input, output, and reusable objects that you want to use in the mapping. Create physical data objects or logical data objects to use as mapping input or output. Create reusable transformations that you want to use. If you want to use mapplets, you must create them also.
3. Create the mapping.
4. Add objects to the mapping. You must add input and output objects to the mapping. Optionally, add transformations and mapplets.
5. Link ports between mapping objects to create a flow of data from sources to targets, through mapplets and transformations that add, remove, or modify data along this flow.
6. Validate the mapping to identify errors.
7. Save the mapping to the Model repository.
After you develop the mapping, run it to see mapping output.

Creating a Mapping

Create a mapping to move data between sources and targets and transform the data.

1. Select a project or folder in the Object Explorer view.
2. Click File > New > Mapping.
3. Enter a mapping name.
4. Click Finish.
   An empty mapping appears in the editor.

Mapping Objects

Mapping objects determine the data flow between sources and targets.

Every mapping must contain the following objects:

- Input. Describes the characteristics of the mapping source.
- Output. Describes the characteristics of the mapping target.

A mapping can also contain the following components:

- Transformation. Modifies data before writing it to targets. Use different transformation objects to perform different functions.
- Mapplet. A reusable object containing a set of transformations that you can use in multiple mappings.

When you add an object to a mapping, you configure the properties according to how you want the Data Integration Service to transform the data. You also connect the mapping objects according to the way you want the Data Integration Service to move the data. You connect the objects through ports.

The editor displays objects in the following ways:

- Iconized. Shows an icon of the object with the object name.
- Normal. Shows the columns and the input and output port indicators. You can connect objects that are in the normal view.

Adding Objects to a Mapping

Add objects to a mapping to determine the data flow between sources and targets.

1. Open the mapping.
2. Drag a physical data object to the editor and select Read to add the data object as a source.
3. Drag a physical data object to the editor and select Write to add the data object as a target.
4. To add a Lookup transformation, drag a flat file data object, logical data object, or relational data object to the editor and select Lookup.
5. To add a reusable transformation, drag the transformation from the Transformations folder in the **Object Explorer** view to the editor.
   Repeat this step for each reusable transformation you want to add.

6. To add a non-reusable transformation, select the transformation on the **Transformation** palette and drag it to the editor.
   Repeat this step for each non-reusable transformation that you want to add.

7. Configure ports and properties for each non-reusable transformation.

8. Optionally, drag a mapplet to the editor.

**Linking Ports**

After you add and configure input, output, transformation, and mapplet objects in a mapping, complete the mapping by linking ports between mapping objects.

Data passes into and out of a transformation through the following ports:

- **Input ports.** Receive data.
- **Output ports.** Pass data.
- **Input/output ports.** Receive data and pass it unchanged.

Every input object, output object, mapplet, and transformation contains a collection of ports. Each port represents a column of data:

- Input objects provide data, so they contain only output ports.
- Output objects receive data, so they contain only input ports.
- Mapplets contain only input ports and output ports.
- Transformations contain a mix of input, output, and input/output ports, depending on the transformation and its application.

To connect ports, you create a link between ports in different mapping objects. The Developer tool creates the connection only when the connection meets link validation and concatenation requirements.

You can leave ports unconnected. The Data Integration Service ignores unconnected ports.

When you link ports between input objects, transformations, mapplets, and output objects, you can create the following types of link:

- One to one
- One to many

You can manually link ports or link ports automatically.

**One to One Links**

Link one port in an input object or transformation to one port in an output object or transformation.
One to Many Links

When you want to use the same data for different purposes, you can link the port providing that data to multiple ports in the mapping.

You can create a one to many link in the following ways:

- Link one port to multiple transformations or output objects.
- Link multiple ports in one transformation to multiple transformations or output objects.

For example, you want to use salary information to calculate the average salary in a bank branch through the Aggregator transformation. You can use the same information in an Expression transformation configured to calculate the monthly pay of each employee.

The following figure shows an example of a mapping with one to many links:

![Image of a mapping with one to many links]

Manually Linking Ports

You can manually link one port or multiple ports.

Drag a port from an input object or transformation to the port of an output object or transformation.

Use the Ctrl or Shift key to select multiple ports to link to another transformation or output object. The Developer tool links the ports, beginning with the top pair. It links all ports that meet the validation requirements.

When you drag a port into an empty port, the Developer tool copies the port and creates a link.

Automatically Linking Ports

When you link ports automatically, you can link by position or by name.

When you link ports automatically by name, you can specify a prefix or suffix by which to link the ports. Use prefixes or suffixes to indicate where ports occur in a mapping.
Linking Ports by Name

When you link ports by name, the Developer tool adds links between input and output ports that have the same name. Link by name when you use the same port names across transformations.

You can link ports based on prefixes and suffixes that you define. Use prefixes or suffixes to indicate where ports occur in a mapping. Link by name and prefix or suffix when you use prefixes or suffixes in port names to distinguish where they occur in the mapping or mapplet.

Linking by name is not case sensitive.

1. Click Mapping > Auto Link.

The Auto Link dialog box appears.

2. Select an object in the From window to link from.
3. Select an object in the To window to link to.
4. Select Name.
5. Optionally, click Show Advanced to link ports based on prefixes or suffixes.
6. Click OK.

Linking Ports by Position

When you link by position, the Developer tool links each output port to the corresponding input port. For example, the first output port is linked to the first input port, the second output port to the second input port. Link by position when you create transformations with related ports in the same order.

1. Click Mapping > Auto Link.

The Auto Link dialog box appears.

2. Select an object in the From window to link from.
3. Select an object in the To window to link to.
4. Select Position and click OK.

The Developer tool links each output port to the corresponding input port. For example, the first output port is linked to the first input port, the second output port to the second input port.

Rules and Guidelines for Linking Ports

Certain rules and guidelines apply when you link ports.

Use the following rules and guidelines when you connect mapping objects:

- If the Developer tool detects an error when you try to link ports between two mapping objects, it displays a symbol indicating that you cannot link the ports.
- Follow the logic of data flow in the mapping. You can link the following types of port:
  - The receiving port must be an input or input/output port.
  - The originating port must be an output or input/output port.
  - You cannot link input ports to input ports or output ports to output ports.
- You must link at least one port of an input group to an upstream transformation.
- You must link at least one port of an output group to a downstream transformation.
- You can link ports from one active transformation or one output group of an active transformation to an input group of another transformation.
- You cannot connect an active transformation and a passive transformation to the same downstream transformation or transformation input group.
- You cannot connect more than one active transformation to the same downstream transformation or transformation input group.
- You can connect any number of passive transformations to the same downstream transformation, transformation input group, or target.
- You can link ports from two output groups in the same transformation to one Joiner transformation configured for sorted data if the data from both output groups is sorted.
- You can only link ports with compatible datatypes. The Developer tool verifies that it can map between the two datatypes before linking them. The Data Integration Service cannot transform data between ports with incompatible datatypes.
- The Developer tool marks some mappings as not valid if the mapping violates data flow validation.

Propagating Port Attributes

Propagate port attributes to pass changed attributes to a port throughout a mapping.

1. In the editor, select a port in a transformation.
2. Click Mapping > Propagate Attributes.
   The Propagate Attributes dialog box appears.
3. Select a direction to propagate attributes.
4. Select the attributes you want to propagate.
5. Optionally, preview the results.
6. Click **Apply**.
   The Developer tool propagates the port attributes.

**Dependency Types**

When you propagate port attributes, the Developer tool updates dependencies.

The Developer tool can update the following dependencies:

- Link path dependencies
- Implicit dependencies

**Link Path Dependencies**

A link path dependency is a dependency between a propagated port and the ports in its link path.

When you propagate dependencies in a link path, the Developer tool updates all the input and input/output ports in its forward link path and all the output and input/output ports in its backward link path. The Developer tool performs the following updates:

- Updates the port name, datatype, precision, scale, and description for all ports in the link path of the propagated port.
- Updates all expressions or conditions that reference the propagated port with the changed port name.
- Updates the associated port property in a dynamic Lookup transformation if the associated port name changes.

**Implicit Dependencies**

An implicit dependency is a dependency within a transformation between two ports based on an expression or condition.

You can propagate datatype, precision, scale, and description to ports with implicit dependencies. You can also parse conditions and expressions to identify the implicit dependencies of the propagated port. All ports with implicit dependencies are output or input/output ports.

When you include conditions, the Developer tool updates the following dependencies:

- Link path dependencies
- Output ports used in the same lookup condition as the propagated port
- Associated ports in dynamic Lookup transformations that are associated with the propagated port
- Master ports used in the same join condition as the detail port

When you include expressions, the Developer tool updates the following dependencies:

- Link path dependencies
- Output ports containing an expression that uses the propagated port

The Developer tool does not propagate to implicit dependencies within the same transformation. You must propagate the changed attributes from another transformation. For example, when you change the datatype of a port that is used in a lookup condition and propagate that change from the Lookup transformation, the Developer tool does not propagate the change to the other port dependent on the condition in the same Lookup transformation.

**Propagated Port Attributes by Transformation**

The Developer tool propagates dependencies and attributes for each transformation.
The following table describes the dependencies and attributes the Developer tool propagates for each transformation:

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Dependency</th>
<th>Propagated Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Expression</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Filter</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Port name</td>
</tr>
<tr>
<td>Joiner</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td>- Implicit Dependencies</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Lookup</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td>- Associated ports (dynamic lookup)</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td>- Implicit Dependencies</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Router</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Port name</td>
</tr>
<tr>
<td>Sorter</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>description</td>
</tr>
<tr>
<td>SQL</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>description</td>
</tr>
<tr>
<td>Union</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Update Strategy</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale,</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>description</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Datatype, precision, scale</td>
</tr>
</tbody>
</table>

Mapping Validation

When you develop a mapping, you must configure it so that the Data Integration Service can read and process the entire mapping. The Developer tool marks a mapping as not valid when it detects errors that will prevent the Data Integration Service from running the mapping.
The Developer tool considers the following types of validation:

- Connection
- Expression
- Object
- Data flow

### Connection Validation

The Developer tool performs connection validation each time you connect ports in a mapping and each time you validate a mapping.

When you connect ports, the Developer tool verifies that you make valid connections. When you validate a mapping, the Developer tool verifies that the connections are valid and that all required ports are connected. The Developer tool makes the following connection validations:

- At least one input object and one output object are connected.
- At least one mapplet input port and output port is connected to the mapping.
- Datatypes between ports are compatible. If you change a port datatype to one that is incompatible with the port it is connected to, the Developer tool generates an error and invalidates the mapping. You can however, change the datatype if it remains compatible with the connected ports, such as Char and Varchar.

### Expression Validation

You can validate an expression in a transformation while you are developing a mapping. If you did not correct the errors, error messages appear in the Validation Log view when you validate the mapping.

If you delete input ports used in an expression, the Developer tool marks the mapping as not valid.

### Object Validation

When you validate a mapping, the Developer tool verifies that the definitions of the independent objects, such as Input transformations or mapplets, match the instance in the mapping.

If any object changes while you configure the mapping, the mapping might contain errors. If any object changes while you are not configuring the mapping, the Developer tool tracks the effects of these changes on the mappings.

### Validating a Mapping

Validate a mapping to ensure that the Data Integration Service can read and process the entire mapping.

1. Click Edit > Validate.
   
   Errors appear in the Validation Log view.
2. Fix errors and validate the mapping again.

### Running a Mapping

Run a mapping to move output from sources to targets and transform data.
If you have not selected a default Data Integration Service, the Developer tool prompts you to select one.

- Right-click an empty area in the editor and click Run Mapping.
  The Data Integration Service runs the mapping and writes the output to the target.

Segments

A segment consists of one or more objects in a mapping, mapplet, or rule. A segment can include a source, target, transformation, or mapplet.

You can copy segments. Consider the following rules and guidelines when you copy a segment:

- You can copy segments across folders or projects.
- The Developer tool reuses dependencies when possible. Otherwise, it copies dependencies.
- If a mapping, mapplet, or rule includes parameters and you copy a transformation that refers to the parameter, the transformation in the target object uses a default value for the parameter.
- You cannot copy input transformations and output transformations.
- After you paste a segment, you cannot undo previous actions.

Copying a Segment

You can copy a segment when you want to reuse a portion of the mapping logic in another mapping, mapplet, or rule.

1. Open the object that contains the segment that you want to copy.
2. Select a segment by highlighting each object you want to copy.
   Hold down the Ctrl key to select multiple objects. You can also select segments by dragging the pointer in a rectangle around objects in the editor.
3. Click Edit > Copy to copy the segment to the clipboard.
4. Open a target mapping, mapplet, or rule.
5. Click Edit > Paste.
This chapter includes the following topics:

- Mapplets Overview, 132
- Mapplet Types, 133
- Mapplets and Rules, 134
- Mapplet Input and Output, 134
- Creating a Mapplet, 135
- Validating a Mapplet, 135

Mapplets Overview

A mapplet is a reusable object containing a set of transformations that you can use in multiple mappings. Use a mapplet in a mapping. Or, validate the mapplet as a rule.

Transformations in a mapplet can be reusable or non-reusable.

When you use a mapplet in a mapping, you use an instance of the mapplet. Any change made to the mapplet is inherited by all instances of the mapplet.

Mapplets can contain other mapplets. You can also use a mapplet more than once in a mapping or mapplet. You cannot have circular nesting of mapplets. For example, if mapplet A contains mapplet B, mapplet B cannot contain mapplet A.
For example, the following figure shows a mapplet that combines customer information from east, west, and central regions with the help of a Union transformation:

The following figure shows a mapping where you can reuse the customer mapplet and find the customer who bought the maximum items in each state:

**Mapplet Types**

The mapplet type is determined by the mapplet input and output.

You can create the following types of mapplet:

- **Source.** The mapplet contains a data source as input and an Output transformation as output.
- **Target.** The mapplet contains an Input transformation as input and a data source as output.
Mapplets and Rules

A rule is business logic that defines conditions applied to source data when you run a profile. It is a midstream mapplet that you use in a profile.

A rule must meet the following requirements:

- It must contain an Input and Output transformation. You cannot use data sources in a rule.
- It can contain Expression transformations and Lookup transformations. It cannot contain any other type of transformation.
- It does not specify cardinality between input groups.

Mapplet Input and Output

To use a mapplet in a mapping, you must configure it for input and output.

A mapplet has the following input and output components:

- Mapplet input. You can pass data into a mapplet from data sources or Input transformations or both. If you validate the mapplet as a rule, you must pass data into the mapplet through an Input transformation. When you use an Input transformation, you connect it to a source or upstream transformation in the mapping.
- Mapplet output. You can pass data out of a mapplet from data sources or Output transformations or both. If you validate the mapplet as a rule, you must pass data from the mapplet through an Output transformation. When you use an Output transformation, you connect it to a target or downstream transformation in the mapping.
- Mapplet ports. You can see mapplet ports in the mapping editor. Mapplet input ports and output ports originate from Input transformations and Output transformations. They do not originate from data sources.

Mapplet Input

Mapplet input can originate from a data source or from an Input transformation.

You can create multiple pipelines in a mapplet. Use multiple data sources or Input transformations. You can also use a combination of data sources and Input transformations.

Use one or more data sources to provide source data in the mapplet. When you use the mapplet in a mapping, it is the first object in the mapping pipeline and contains no input ports.

Use an Input transformation to receive input from the mapping. The Input transformation provides input ports so you can pass data through the mapplet. Each port in the Input transformation connected to another transformation in the mapplet becomes a mapplet input port. Input transformations can receive data from a single active source. Unconnected ports do not appear in the mapping editor.

You can connect an Input transformation to multiple transformations in a mapplet. You can also connect one port in an Input transformation to multiple transformations in the mapplet.
Mapplet Output

Use a data source as output when you want to create a target mapplet. Use an Output transformation in a mapplet to pass data through the mapplet into a mapping.

Use one or more data sources to provide target data in the mapplet. When you use the mapplet in a mapping, it is the last object in the mapping pipeline and contains no output ports.

Use an Output transformation to pass output to a downstream transformation or target in a mapping. Each connected port in an Output transformation appears as a mapplet output port in a mapping. Each Output transformation in a mapplet appears as an output group. An output group can pass data to multiple pipelines in a mapping.

Creating a Mapplet

Create a mapplet to define a reusable object containing a set of transformations that you can use in multiple mappings.

1. Select a project or folder in the Object Explorer view.
2. Click File > New > Mapplet.
3. Enter a mapplet name.
4. Click Finish.
   An empty mapplet appears in the editor.
5. Add mapplet inputs, outputs, and transformations.

Validating a Mapplet

Validate a mapplet before you add it to a mapping. You can also validate a mapplet as a rule to include it in a profile.

1. Right-click the mapplet editor.
2. Select Validate As > Mapplet or Validate As > Rule.
   The Validation Log displays mapplet error messages.
Viewing Data Overview

You can run a mapping, view profile results, view source data, or preview data for a transformation.

Run a mapping to move output from sources to targets and transform data. You can run a mapping from the command line or from the Run dialog box. View profile results in the editor.

You view source data or preview data for a transformation in the Data Viewer view. Before you can view data, you need to select the default Data Integration Service. You can create configurations to control settings that the Developer tool applies when you view data.

When you view data in the Data Viewer view, you can export the data to a file. You can also access logs that show log events.

Configurations

A configuration is a group of settings that the Developer tool applies when you run a mapping or preview data.

A configuration controls settings such as the default Data Integration Service, number of rows to read from a source, default date/time format, and optimizer level. The configurations that you create apply to your installation of the Developer tool.

You can create the following configurations:

- Data viewer configurations. Control the settings the Developer tool applies when you preview output in the Data Viewer view.
- Mapping configurations. Control the settings the Developer tool applies when you run mappings through the Run Configurations dialog box or from the command line.
Configuration Properties

The Developer tool applies configuration properties when you preview output or you run mappings. Set configuration properties for the Data Viewer view or mappings in the Run dialog box.

Data Integration Service Properties

The Developer tool displays the Data Integration Service tab for data viewer and mapping configurations. The following table displays the properties that you configure for the Data Integration Service:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use default Data Integration Service</td>
<td>Uses the default Data Integration Service to run the mapping. Default is enabled.</td>
</tr>
<tr>
<td>Data Integration Service</td>
<td>Specifies the Data Integration Service that runs the mapping if you do not use the default Data Integration Service.</td>
</tr>
</tbody>
</table>

Source Properties

The Developer tool displays the Source tab for data viewer and mapping configurations. The following table displays the properties that you configure for sources:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read all rows</td>
<td>Reads all rows from the source. Default is enabled.</td>
</tr>
<tr>
<td>Read up to how many rows</td>
<td>Specifies the maximum number of rows to read from the source if you do not read all rows. Note: If you enable this option for a mapping that writes to a customized data object, the Data Integration Service does not truncate the target table before it writes to the target. Default is 1000.</td>
</tr>
<tr>
<td>Read all characters</td>
<td>Reads all characters in a column. Default is disabled.</td>
</tr>
<tr>
<td>Read up to how many characters</td>
<td>Specifies the maximum number of characters to read in each column if you do not read all characters. The Data Integration Service ignores this property for SAP sources. Default is 4000.</td>
</tr>
</tbody>
</table>

Results Properties

The Developer tool displays the Results tab for data viewer configurations. The following table displays the properties that you configure for results in the Data Viewer view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show all rows</td>
<td>Displays all rows in the Data Viewer view.</td>
</tr>
</tbody>
</table>
### Property Description

- **Default is disabled.**

- **Show up to how many rows**
  Specifies the maximum number of rows to display if you do not display all rows.
  Default is 1000.

- **Show all characters**
  Displays all characters in a column.
  Default is disabled.

- **Show up to how many characters**
  Specifies the maximum number of characters to display in each column if you do not display all characters.
  Default is 4000.

---

### Advanced Properties

The Developer tool displays the Advanced tab for data viewer and mapping configurations.

The following table displays the advanced properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default date time format</td>
<td>Date/time format the Data Integration Services uses when the mapping converts strings to dates. Default is MM/DD/YYYY HH24:MI:SS.</td>
</tr>
</tbody>
</table>
| Override tracing level  | Overrides the tracing level for each transformation in the mapping. The tracing level determines the amount of information that the Data Integration Service sends to the mapping log files. Choose one of the following tracing levels:  
  - None. The Data Integration Service uses the tracing levels set in the mapping.  
  - Terse. The Data Integration Service logs initialization information, error messages, and notification of rejected data.  
  - Normal. The Data Integration Service logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes mapping results, but not at the level of individual rows.  
  - Verbose initialization. In addition to normal tracing, the Data Integration Service logs additional initialization details, names of index and data files used, and detailed transformation statistics.  
  - Verbose data. In addition to verbose initialization tracing, the Data Integration Service logs each row that passes into the mapping. Also notes where the Data Integration Service truncates string data to fit the precision of a column and provides detailed transformation statistics.  
  Default is None. |
| Sort order             | Order in which the Data Integration Service sorts character data in the mapping. Default is Binary. |
### Property Description

**Optimizer level**
- Controls the optimization methods that the Data Integration Service applies to a mapping as follows:
  - None. The Data Integration Service does not optimize the mapping.
  - Minimal. The Data Integration Service applies the early projection optimization method to the mapping.
  - Normal. The Data Integration Service applies the early projection, early selection, pushdown, and predicate optimization methods to the mapping.
  - Full. The Data Integration Service applies the early projection, early selection, pushdown, predicate, cost-based, and semi-join optimization methods to the mapping.
  
  Default is Normal.

**High precision**
- Runs the mapping with high precision.
  
  High precision data values have greater accuracy. Enable high precision if the mapping produces large numeric values, for example, values with precision of more than 15 digits, and you require accurate values. Enabling high precision prevents precision loss in large numeric values.
  
  Default is enabled.

**Send log to client**
- Allows you to view log files in the Developer tool. If you disable this option, you must view log files through the Administrator tool.
  
  Default is enabled.

---

**Data Viewer Configurations**

Data viewer configurations control the settings that the Developer tool applies when you preview output in the **Data Viewer** view.

You can select a data viewer configuration when you preview output for the following objects:

- Custom data objects
- Logical data objects
- Logical data object read mappings
- Physical data objects
- Sources and transformations within mappings

**Creating a Data Viewer Configuration**

Create a data viewer configuration to control the settings the Developer tool applies when you preview output in the **Data Viewer** view.
1. Click Run > Open Run Dialog.
   The Run Configurations dialog box appears.

   ![Run Configurations dialog box]

2. Click Data Viewer Configuration.

3. Click the New button ( ).
The right panel of the **Run Configurations** dialog box displays the data viewer configuration properties.

4. Enter a name for the data viewer configuration.
5. Configure the data viewer configuration properties.
6. Click **Apply**.
7. Click **Close**.

The Developer tool creates the data viewer configuration.

**Mapping Configurations**

Mapping configurations control the mapping deployment properties that the Developer tool uses when you run a mapping through the **Run Configurations** dialog box or from the command line.

To apply a mapping configuration to a mapping that you run through the Developer tool, you must run the mapping through the **Run Configurations** dialog box. If you run the mapping through the **Run** menu or mapping editor, the Developer tool runs the mapping with the default mapping deployment properties.

To apply mapping deployment properties to a mapping that you run from the command line, select the mapping configuration when you add the mapping to an application. The mapping configuration that you select applies to all mappings in the application.

You can change the mapping deployment properties when you edit the application. An administrator can also change the mapping deployment properties through the Administrator tool. You must redeploy the application for the changes to take effect.

**Creating a Mapping Configuration**

Create a mapping configuration to control the mapping deployment properties that the Developer tool uses when you run mappings through the **Run** dialog box or from the command line.
1. Click **Run > Open Run Dialog**.

   The **Run Configurations** dialog box appears.

2. Click **Mapping Configuration**.

3. Click the **New button** ( ).
The right panel of the Run Configurations dialog box displays the mapping configuration properties.

4. Enter a name for the mapping configuration.
5. Configure the mapping configuration properties.
6. Click **Apply**.
7. Click **Close**.
   
The Developer tool creates the mapping configuration.

### Updating the Default Configuration Properties

You can update the default data viewer and mapping configuration properties.

1. Click **Window > Preferences**.
   
The Preferences dialog box appears.
2. Click **Informatica > Run Configurations**.
3. Select the Data Viewer or Mapping configuration.
4. Configure the default data viewer or mapping configuration properties.

5. Click **OK**.

The Developer tool updates the default configuration properties.

**Troubleshooting Configurations**

I created two configurations with the same name but with different cases. When I close and reopen the Developer tool, one configuration is missing.

Data viewer and mapping configuration names are not case sensitive. If you create multiple configurations with the same name but different cases, the Developer tool deletes one of the configurations when you exit. The Developer tool does not consider the configuration names unique.

I tried to create a configuration with a long name, but the Developer tool displays an error message that says it cannot write the file.

The Developer tool stores data viewer and mapping configurations in files on the machine that runs the Developer tool. If you create a configuration with a long name, for example, more than 100 characters, the Developer tool might not be able to save the file to the hard drive.

To work around this issue, shorten the configuration name.
Exporting Data

You can export the data that appears in the Data Viewer view to a tab-delimited flat file, such as a TXT or CSV file. Export data when you want to create a local copy of the data.

1. In the Data Viewer view, right-click the results and select Export Data.
2. Enter a file name and extension.
3. Select the location where you want to save the file.
4. Click OK.

Logs

The Data Integration Service generates log events when you run a mapping, run a profile, or preview data. Log events include information about the tasks performed by the Data Integration Service, errors, and load summary and transformation statistics.

You can view the logs generated from the Developer tool and save them to a local directory.

You can view log events from the Show Log button in the Data Viewer view.

The following figure shows the Show Log button in the Data Viewer view:

When you run a mapping from Run > Run Mapping, you can view the log events from the Progress view. To open the log events in the Developer tool, click the link for the mapping run and select Go to Log.
The following figure shows the link for a mapping run in the **Progress** view:

![Progress view](image)

When you run a profile, you can view the log events from the Monitoring tool.

To save the log to a file, click **File > Save a Copy As** and choose a directory. By default the log files are stored in the following directory: `c:\[TEMP]\AppData\Local\Temp`.

### Log File Format

The information in the log file depends on the sequence of events during the run. The amount of information that is sent to the logs depends on the tracing level.

The Data Integration Service updates log files with the following information when you run a mapping, run a profile, or preview data:

- **Logical DTM messages**
  - Contain information about preparing to compile, to optimize, and to translate the mapping. The log events and the amount of information depends on the configuration properties set.

- **Data Transformation Manager (DTM) messages**
  - Contain information about establishing a connection to the source, reading the data, transforming the data, and loading the data to the target.

- **Load summary and transformation statistics messages**
  - Contain information about the number of rows read from the source, number of rows output to the target, number of rows rejected, and the time to execute.

### Monitoring Jobs from the Developer Tool

You can access the Monitoring tool from the Developer tool to monitor the status of applications and jobs, such as a profile jobs. As an administrator, you can also monitor applications and jobs in the Administrator tool.

Monitor applications and jobs to view properties, run-time statistics, and run-time reports about the integration objects. For example, you can see the general properties and the status of a profiling job. You can also see who initiated the job and how long it took the job to complete.

To monitor applications and jobs from the Developer tool, click the Menu button in the **Progress** view and select **Monitor Jobs**. Select the Data Integration Service that runs the applications and jobs and click **OK**. The Monitoring tool opens.
Chapter 12

Workflows

This chapter includes the following topics:

- Workflows Overview, 147
- Creating a Workflow, 148
- Workflow Objects, 148
- Sequence Flows, 150
- Workflow Advanced Properties, 152
- Workflow Validation, 153
- Workflow Deployment, 154
- Running Workflows, 155
- Monitoring Workflows, 155
- Deleting a Workflow, 155
- Workflow Examples, 156

Workflows Overview

A workflow is a graphical representation of a set of events, tasks, and decisions that define a business process. You use the Developer tool to add objects to a workflow and to connect the objects with sequence flows. The Data Integration Service uses the instructions configured in the workflow to run the objects.

A workflow object is an event, task, or gateway. An event starts or ends the workflow. A task is an activity that runs a single unit of work in the workflow, such as running a mapping, sending an email, or running a shell command. A gateway makes a decision to split and merge paths in the workflow.

A sequence flow connects workflow objects to specify the order that the Data Integration Service runs the objects. You can create a conditional sequence flow to determine whether the Data Integration Service runs the next object.

You can define and use workflow variables and parameters to make workflows more flexible. A workflow variable represents a value that records run-time information and that can change during a workflow run. A workflow parameter represents a constant value that you define before running a workflow. You use workflow variables and parameters in conditional sequence flows and object fields. You also use workflow variables and parameters to pass data between a task and the workflow.

To develop a workflow, complete the following steps:

1. Create a workflow.
2. Add objects to the workflow and configure the object properties.
3. Connect objects with sequence flows to specify the order that the Data Integration Service runs the objects. Create conditional sequence flows to determine whether the Data Integration Service runs the next object.

4. Define variables for the workflow to capture run-time information. Use the workflow variables in conditional sequence flows and object fields.

5. Define parameters for the workflow so that you can change parameter values each time you run a workflow. Use the workflow parameters in conditional sequence flows and object fields.

6. Validate the workflow to identify errors.

7. Add the workflow to an application and deploy the application to the Data Integration Service.

After you deploy a workflow, you run an instance of the workflow from the deployed application using the infacmd wfs command line program. You monitor the workflow instance run in the Monitoring tool.

For more information, see the PowerCenter Express Workflow Guide.

Creating a Workflow

When you create a workflow, the Developer tool adds a Start event and an End event to the workflow.

1. Select a project or folder in the Object Explorer view.

2. Click File > New > Workflow.

   The Developer tool gives the workflow a default name.

3. Optionally, edit the default workflow name.

4. Click Finish.

   A workflow with a Start event and an End event appears in the editor.

Workflow Objects

A workflow object is an event, task, or gateway. You add objects as you develop a workflow in the editor. Workflow objects are non-reusable. The Developer tool stores the objects within the workflow only.

Events

An event starts or ends the workflow. An event represents something that happens when the workflow runs. The editor displays events as circles.

The following table describes all events that you can add to a workflow:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Represents the beginning of the workflow. A workflow must contain one Start event.</td>
</tr>
<tr>
<td>End</td>
<td>Represents the end of the workflow. A workflow must contain one End event.</td>
</tr>
</tbody>
</table>

The Developer tool gives each event a default name of Start_Event or End_Event. You can rename and add a description to an event in the event properties.
Tasks

A task in an activity that runs a single unit of work in the workflow, such as running a mapping, sending an email, or running a shell command. A task represents something that is performed during the workflow. The editor displays tasks as squares.

The following table describes all tasks that you can add to a workflow:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Assigns a value to a user-defined workflow variable.</td>
</tr>
<tr>
<td>Command</td>
<td>Runs a single shell command or starts an external executable program.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Runs a mapping.</td>
</tr>
<tr>
<td>Notification</td>
<td>Sends an email notification to specified recipients.</td>
</tr>
</tbody>
</table>

A workflow can contain multiple tasks of the same task type.

The Developer tool gives each task a default name of `<task type> Task`, for example `Command Task`. When you add another task of the same type to the same workflow, the Developer tool appends an integer to the default name, for example `Command_Task1`. You can rename and add a description to a task in the task general properties.

Exclusive Gateways

An Exclusive gateway splits and merges paths in the workflow based on how the Data Integration Service evaluates expressions in conditional sequence flows. An Exclusive gateway represents a decision made in the workflow. The editor displays Exclusive gateways as diamonds.

When an Exclusive gateway splits the workflow, the Data Integration Service makes a decision to take one of the outgoing branches. When an Exclusive gateway merges the workflow, the Data Integration Service waits for one incoming branch to complete before triggering the outgoing branch.

When you add an Exclusive gateway to split a workflow, you must add another Exclusive gateway to merge the branches back into a single flow.

The Developer tool gives each Exclusive gateway a default name of `Exclusive_Gateway`. When you add another Exclusive gateway to the same workflow, the Developer tool appends an integer to the default name, for example `Exclusive_Gateway1`. You can rename and add a description to an Exclusive gateway in the gateway general properties.

Adding Objects to a Workflow

Add the tasks and gateways that you want to run in the workflow. A workflow must contain one Start event and one End event. When you create a workflow, the Developer tool adds the Start event and End event to the workflow.

1. Open the workflow in the editor.
2. Select an object from the Workflow Object palette and drag it to the editor. If you selected a Mapping task, click Browse to select the mapping and then click Finish.
   Or to add a Mapping task, select a mapping from the Object Explorer view and drag it to the editor.
   The object appears in the editor. Select the object to configure the object properties.
Sequence Flows

A sequence flow connects workflow objects to specify the order that the Data Integration Service runs the objects. The editor displays sequence flows as arrows. You can create conditional sequence flows to determine whether the Data Integration Service runs the next object.

You cannot use sequence flows to create loops. Each sequence flow can run one time.

The number of incoming and outgoing sequence flows that an object can have depends on the object type:

**Events**

A Start event must have a single outgoing sequence flow. An End event must have a single incoming sequence flow.

**Tasks**

Tasks must have a single incoming sequence flow and a single outgoing sequence flow.

**Gateways**

Gateways must have either multiple incoming sequence flows or multiple outgoing sequence flows, but not both. Use multiple outgoing sequence flows from an Exclusive gateway to split a workflow. Use multiple incoming sequence flows to an Exclusive gateway to merge multiple branches into a single flow.

When you connect objects, the Developer tool gives the sequence flow a default name. The Developer tool names sequence flows using the following format:

```
<originating object name>_to_<ending object name>
```

If you create a conditional sequence flow, you might want to rename the sequence flow to indicate the conditional expression. For example, if a conditional sequence flow from a Mapping task to a Command task includes a condition that checks if the Mapping task ran successfully, you might want to rename the sequence flow to MappingSucceeded. You can rename and add a description to a sequence flow in the sequence flow general properties.

Conditional Sequence Flows

Create a conditional sequence flow to determine whether the Data Integration Service runs the next object in the workflow.

A conditional sequence flow includes an expression that the Data Integration Service evaluates to true or false. The expression must return either a boolean or an integer value. If an expression returns an integer value, any non-zero value is the equivalent of true. A value of zero (0) is the equivalent of false.

If the expression evaluates to true, the Data Integration Service runs the next object. If the expression evaluates to false, the Data Integration Service does not run the next object. If you do not specify a condition in a sequence flow, the Data Integration Service runs the next object by default.

When an expression in a conditional sequence flow evaluates to false, the Data Integration Service does not run the next object or any of the subsequent objects in that branch. When you monitor the workflow, the Monitoring tool does not list objects that do not run in the workflow. When a workflow includes objects that do not run, the workflow can still complete successfully.

You cannot create a conditional sequence flow from the Start event to the next object in the workflow or from the last object in the workflow to the End event.
Failed Tasks and Conditional Sequence Flows

By default, the Data Integration Service continues to run subsequent objects in a workflow after a task fails. To stop running subsequent workflow objects after a task fails, use a conditional sequence flow that checks if the previous task succeeds.

You can use a conditional sequence flow to check if a Mapping, Command, or Notification task succeeds. These tasks return an Is Successful general output. The Is Successful output contains true if the task ran successfully, or it contains false if the task failed. Create a boolean workflow variable that captures the Is Successful output returned by a task. Then, create an expression in the outgoing conditional sequence flow that checks if the variable value is true.

For example, you create a boolean workflow variable that captures the Is Successful output returned by a Mapping task. You create the following expression in the conditional sequence flow that connects the Mapping task to the next task in the workflow:

\[
\text{$\var: MappingTaskSuccessful = true$}
\]

If the Mapping task fails, the expression evaluates to false and the Data Integration Service stops running any subsequent workflow objects.

Parameters and Variables in Conditional Sequence Flows

You can include workflow parameters and variables in an expression for a conditional sequence flow.

You can select a workflow parameter or variable in the **Condition** tab, or you can type the parameter or variable name in the conditional expression using the required syntax.

For example, you create a workflow variable that captures the number of rows written to the target by a mapping run by a Mapping task. You create the following expression in the conditional sequence flow that connects the Mapping task to a Command task:

\[
\text{$\var: TargetRowsMapping > 500$}
\]

The Data Integration Service runs the Command task if the mapping wrote more than 500 rows to the target.

Connecting Objects

Connect objects with sequence flows to determine the order that the Data Integration Service runs the objects in the workflow.

To connect two objects, select the first object in the editor and drag it to the second object. To connect multiple objects, use the **Connect Workflow Objects** dialog box.

1. Right-click in the editor and select **Connect Workflow Objects**.
2. Select the object that you want to connect from, select the object that you want to connect to, and click **Apply**.
3. Continue connecting additional objects, and then click **OK**.

The sequence flows appear between the objects.

Creating a Conditional Sequence Flow

A conditional sequence flow includes an expression that evaluates to true or false. Create a conditional sequence flow to determine whether the Data Integration Service runs the next object in the workflow.

1. Select a sequence flow in the editor.
2. In the **Properties** view, click the **Condition** tab.
3. Enter the conditional expression.
   The Functions tab lists transformation language functions. The Inputs tab lists workflow parameters and variables. Double-click a function, parameter, or variable name to include it in the expression. Type operators and literal values into the expression as needed.

4. Validate the condition using the Validate button.
   Errors appear in a dialog box.

5. If an error appears, fix the error and validate the condition again.

Workflow Advanced Properties

The workflow advanced properties include properties that define how workflow instances run.

Tracing Level

Determines the amount of detail that appears in the workflow log. You can select a value for the tracing level. Or, you can assign the tracing level to a parameter so that you can define the value of the property in a workflow parameter. The tracing level has a string datatype.

Default is INFO.

The following table describes the workflow tracing levels:

<table>
<thead>
<tr>
<th>Tracing Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>Logs error messages that caused the workflow instance to fail. The workflow log displays this level as SEVERE.</td>
</tr>
<tr>
<td>WARNING</td>
<td>In addition to the error level messages, logs warning messages that indicate failures occurred, but the failures did not cause the workflow instance to fail. The workflow log displays this level as WARNING.</td>
</tr>
<tr>
<td>INFO</td>
<td>In addition to the warning level messages, logs additional initialization information and details about the workflow instance run. Logs task processing details including the input data passed to the task, the work item completed by the task, and the output data produced by the task. Also logs the parameter file name and expression evaluation results for conditional sequence flows. The workflow log displays this level as INFO.</td>
</tr>
<tr>
<td>TRACE</td>
<td>In addition to the info level messages, logs additional details about workflow or task initialization. The workflow log displays this level as FINE.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>In addition to the trace level messages, logs additional details about task input and task output and about the workflow state. The workflow log displays this level as FINEST.</td>
</tr>
</tbody>
</table>
Workflow Validation

When you develop a workflow, you must configure it so that the Data Integration Service can read and process the entire workflow. The Developer tool marks a workflow as not valid when it detects errors that will prevent the Data Integration Service from running the workflow.

When you validate a workflow, the Developer tool validates sequence flows, expressions, and workflow objects.

Sequence Flow Validation

The Developer tool performs sequence flow validation each time you validate a workflow.

The Developer tool makes the following sequence flow validations:

- The workflow cannot run if the sequence flows loop. Each sequence flow can run one time.
- The Start event has one outgoing sequence flow that does not include a condition.
- The End event has one incoming sequence flow.
- Each task has one incoming sequence flow and one outgoing sequence flow.
- Each Exclusive gateway has either multiple incoming sequence flows or multiple outgoing sequence flows, but not both. Each Exclusive gateway that splits the workflow has at least two outgoing sequence flows with one of the sequence flows set as the default. Each Exclusive gateway that merges the workflow does not have a default outgoing sequence flow.
- For a conditional sequence flow, the expression returns a boolean or integer value. The expression cannot contain a carriage return character or line feed character.

Expression Validation

You can validate an expression in a conditional sequence flow or in an Assignment task while you are creating the expression. If you did not correct the errors, error messages appear in the Validation Log view when you validate the workflow.

Workflow Object Validation

The Developer tool performs workflow object validation each time you validate a workflow.

The Developer tool validates the following workflow objects:

Events

The workflow contains one Start event that is the first object in the workflow. The workflow contains one End event that is the last object in the workflow. The workflow has a path from the Start event to the End event.

Tasks

Each task has a unique name within the workflow. If applicable, task input is assigned to workflow parameters and variables with compatible types. If applicable, task output is assigned to workflow variables with compatible datatypes. Task configuration properties are assigned to valid values.

Each Assignment task assigns a valid value to a single workflow variable. The value assigned to the workflow variable has a compatible datatype. If the task uses workflow parameters or variables in the assignment expression, the Developer tool verifies that the parameters and variables exist.

Each Command task includes a command that does not contain a carriage return character or line feed character. If the command uses workflow parameters or variables, the Developer tool verifies that the parameters and variables exist.
Each Mapping task includes a valid mapping that exists in the repository.

Each Notification task includes at least one recipient. If the task uses workflow parameters or variables, the Developer tool verifies that the parameters and variables exist.

**Gateways**

Each Exclusive gateway has a unique name within the workflow.

**Validating a Workflow**

Validate a workflow to ensure that the Data Integration Service can read and process the entire workflow.

1. Open the workflow in the editor.
2. Click **Edit > Validate**.
   
   Errors appear in the **Validation Log** view.
3. If an error appears, fix the error and validate the workflow again.

**Workflow Deployment**

When you develop a workflow in the Developer tool, you create a workflow definition. To run an instance of the workflow, you add the workflow definition to an application. Then, you deploy the application to the Data Integration Service.

Deploy workflows to allow users to run workflows using the infacmd wfs startWorkflow command. When you deploy a workflow, the Data Integration Service creates a separate set of run-time metadata in the Model repository for the workflow. If you make changes to a workflow definition in the Developer tool after you deploy it, you must redeploy the application that contains the workflow definition for the changes to take effect.

Use the Developer tool to deploy workflows. You deploy workflows using the same procedure that you use to deploy other Model repository objects.

**Deploy and Run a Workflow**

When you deploy a workflow to the Data Integration Service, you can run a single instance of the workflow immediately after you deploy it. When you deploy and run a workflow, you cannot specify a parameter file. If the workflow uses parameters, the Data Integration Service uses the default parameter values.

To run a workflow immediately after you deploy it, click **Run Object** in the **Deploy Completed** dialog box. If the deployed application contains multiple workflows, select the workflows to run. The Data Integration Service concurrently runs an instance of each selected workflow. If the deployed application contains other object types, you cannot select those objects to run.

Monitor the workflow instance run in the Monitoring tab of the Administrator tool. To run additional instances of the workflow, use the infacmd wfs startWorkflow command.

If you receive an error message when you deploy and run a workflow, view the workflow and Data Integration Service logs for more information.
Running Workflows

After you deploy a workflow, you run an instance of the workflow from the deployed application using the infacmd wfs startWorkflow command. You can specify a parameter file for the workflow run.

You can concurrently run multiple instances of the same workflow from the deployed application. When you run a workflow instance, the application sends the request to the Data Integration Service. The Data Integration Service runs the objects in the workflow according to the sequence flows connecting the objects.

For example, the following command runs an instance of the workflow MyWorkflow in the deployed application MyApplication using the parameter values defined in the parameter file MyParameterFile:

```
```

Monitoring Workflows

You monitor a workflow instance run in the Monitoring tool. The Monitoring tool is a direct link to the Monitoring tab of the Administrator tool.

The Monitoring tool shows the status of running workflow and workflow object instances. You can abort or cancel a running workflow instance in the Monitoring tool. You can also use the Monitoring tool to view logs for workflow instances and to view workflow reports.

Deleting a Workflow

You might decide to delete a workflow that you no longer use. When you delete a workflow, you delete all objects in the workflow.

When you delete a workflow in the Developer tool, you delete the workflow definition in the Model repository. If the workflow definition has been deployed to a Data Integration Service, you can continue to run instances of the workflow from the deployed workflow definition.

To delete a workflow, select the workflow in the Object Explorer view and then click Edit > Delete.
Workflow Examples

The following examples show how you might want to develop workflows.

**Example: Running Commands Before and After Running a Mapping**

You can develop a workflow that runs commands to perform steps before and after a mapping runs. For example, you might want to use Command tasks before and after a Mapping task to drop indexes on the target before the mapping runs, and then recreate the indexes when the mapping completes.

The following figure shows a workflow that runs a command, runs a mapping, runs another command, and sends an email notifying users of the status of the workflow:

Parameter files provide you with the flexibility to change parameter values each time you run a workflow. You can use the following parameters in this workflow:

- Workflow parameter that represents the command run by the first Command task.
- Mapping parameter that represents the connection to the source for the mapping.
- Mapping parameter that represents the connection to the target for the mapping.
- Workflow parameter that represents the command run by the second Command task.
- Workflow parameter that represents the email address that the Notification task sends an email to.

Define the value of these parameters in a parameter file. Specify the parameter file when you run the workflow. You can run the same workflow with a different parameter file to run different commands, to run a mapping that connects to a different source or target, or to send an email to a different user.

**Example: Splitting a Workflow**

You can develop a workflow that includes an Exclusive gateway that makes a decision to split the workflow.

For example, you can develop the following workflow that runs a mapping, decides to take one branch of the workflow depending on whether the Mapping task succeeded or failed, merges the branches back into a single flow, and sends an email notifying users of the status of the workflow:

This workflow includes the following components:

- Mapping task that runs a mapping and then assigns the Is Successful output to a boolean workflow variable.
Exclusive gateway that includes two outgoing sequence flows. One sequence flow includes a condition that evaluates the value of the workflow variable. If the condition evaluates to true, the Data Integration Service runs the connected task. If the condition evaluates to false, the Data Integration Service takes the other branch.

Two workflow branches that can include any number of tasks. In this example, each branch includes a different command, mapping, and another command. The Data Integration Service takes one of these branches.

Exclusive gateway that merges the two branches back into a single flow.

Notification task that sends an email notifying users of the status of the workflow.
CHAPTER 13

Deployment

This chapter includes the following topics:

- Deployment Overview, 158
- Deployment Methods, 159
- Mapping Deployment Properties, 159
- Creating an Application, 160
- Deploying an Object to a Data Integration Service, 161
- Deploying an Object to a File, 161
- Updating an Application, 162
- Importing Application Archives, 162
- Application Redeployment, 162

Deployment Overview

Deploy objects to make them accessible to end users. You can deploy physical data objects, logical data objects, mappings, mapplets, workflows, and applications.

Deploy objects to allow users to query the objects through a third-party client tool or to run mappings or workflows at the command line. When you deploy an object, you isolate the object from changes in data structures. If you make changes to an object in the Developer tool after you deploy it, you must redeploy the application that contains the object for the changes to take effect.

You can deploy objects to a Data Integration Service or a network file system. When you deploy an application to a Data Integration Service, end users can connect to the application. Depending on the types of objects in the application, end users can then run queries against the objects, or run mappings or workflows. The end users must have the appropriate permissions in the Administrator tool to perform these tasks.

When you deploy an object to a network file system, the Developer tool creates an application archive file. Deploy an object to a network file system if you want to check the application into a version control system. You can also deploy an object to a file if your organization requires that administrators deploy objects to Data Integration Services. An administrator can deploy application archive files to Data Integration Services through the Administrator tool. You can also import objects from an application archive into projects or folders in the Model repository.
Deployment Methods

Deploy objects or deploy an application that contains one or more objects. The object deployment method differs based on the type of object that you deploy.

**Deploy an object**

Deploy an object to make the object available to end users. If you redeploy an object to a Data Integration Service, you cannot update the application. The Developer tool creates an application with a different name.

When you deploy the following objects, the Developer tool prompts you to create an application and the Developer tool adds the object to the application:

- Mappings
- Workflows

**Deploy an application that contains objects**

Create an application to deploy multiple objects at the same time. When you create an application, you select the objects to include in the application. If you redeploy an application to a Data Integration Service you can update or replace the application.

Mapping Deployment Properties

When you update an application that contains a mapping, you can set the deployment properties that the Data Integration Service uses when end users run the mapping.

Set mapping deployment properties on the **Advanced** view of the application.

You can set the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default date time format</td>
<td>Date/time format that the Data Integration Service uses when the mapping converts strings to dates. Default is MM/DD/YYYY HH24:MI:SS.</td>
</tr>
<tr>
<td>Override tracing level</td>
<td>Overrides the tracing level for each transformation in the mapping. The tracing level determines the amount of information that the Data Integration Service sends to the mapping log files.</td>
</tr>
<tr>
<td></td>
<td>Choose one of the following tracing levels:</td>
</tr>
<tr>
<td></td>
<td>- None. The Data Integration Service does not override the tracing level that you set for each transformation.</td>
</tr>
<tr>
<td></td>
<td>- Terse. The Data Integration Service logs initialization information, error messages, and notification of rejected data.</td>
</tr>
<tr>
<td></td>
<td>- Normal. The Data Integration Service logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. It summarizes mapping results, but not at the level of individual rows.</td>
</tr>
<tr>
<td></td>
<td>- Verbose Initialization. In addition to normal tracing, the Data Integration Service logs additional initialization details, names of index and data files used, and detailed transformation statistics.</td>
</tr>
<tr>
<td></td>
<td>- Verbose Data. In addition to verbose initialization tracing, the Data Integration Service logs each row that passes into the mapping. The Data Integration Service also notes where it truncates string data to fit the precision of a column and provides detailed transformation statistics. The Data Integration Service writes row data for all rows in a block when it processes a transformation.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default is None.</td>
<td></td>
</tr>
<tr>
<td>Sort order</td>
<td>Order in which the Data Integration Service sorts character data in the mapping. Default is Binary.</td>
</tr>
</tbody>
</table>
| Optimizer level | Controls the optimization methods that the Data Integration Service applies to a mapping as follows:  
- None. The Data Integration Service does not optimize the mapping.  
- Minimal. The Data Integration Service applies the early projection optimization method to the mapping.  
- Normal. The Data Integration Service applies the early projection, early selection, pushdown, and predicate optimization methods to the mapping.  
- Full. The Data Integration Service applies the early projection, early selection, pushdown, predicate, cost-based, and semi-join optimization methods to the mapping. Default is Normal. |
| High precision | Runs the mapping with high precision. High precision data values have greater accuracy. Enable high precision if the mapping produces large numeric values, for example, values with precision of more than 15 digits, and you require accurate values. Enabling high precision prevents precision loss in large numeric values. Default is enabled. |

Creating an Application

Create an application when you want to deploy multiple objects at the same time or if you want to be able to update or replace the application when it resides on the Data Integration Service. When you create an application, you select the objects to include in the application.

1. Select a project or folder in the Object Explorer view.
3. Enter a name for the application.
4. Click Browse to select the application location. You must create the application in a project or a folder.
5. Click Next. The Developer tool prompts you for the objects to include in the application.
6. Click Add. The Add Objects dialog box appears.
7. Select one or more objects and click OK. The Developer tool lists the objects you select in the New Application dialog box.
8. If the application contains mappings, choose whether to override the default mapping configuration when you deploy the application. If you select this option, choose a mapping configuration. The Developer tool sets the mapping deployment properties for the application to the same values as the settings in the mapping configuration.
9. Click Finish.
The Developer tool adds the application to the project or folder.

After you create an application, you must deploy the application so end users can run mappings or workflows.

Deploying an Object to a Data Integration Service

Deploy an object to a Data Integration Service so end users can run mappings or workflows from the command line.

1. Right-click an object in the Object Explorer view and select Deploy.
The Deploy dialog box appears.
2. Select Deploy to Service.
3. Click Browse to select the domain.
The Choose Domain dialog box appears.
4. Select a domain and click OK.
The Developer tool lists the Data Integration Service associated with the domain in the Available Services section of the Deploy Application dialog box.
5. Select the Data Integration Service to which you want to deploy the application. Click Next.
6. Enter an application name.
7. Click Finish.
The Developer tool deploys the application to the Data Integration Service.

Deploying an Object to a File

Deploy an object to an application archive file if you want to check the application into version control or if you want to deploy objects to the Data Integration Service.

1. Right-click an object in the Object Explorer view and select Deploy.
The Deploy dialog box appears.
2. Select Deploy to File System.
3. Click Browse to select the directory.
The Choose a Directory dialog box appears.
4. Select the directory and click OK. Then, click Next.
5. Enter an application name.
6. Click Finish.
The Developer tool deploys the application to an application archive file.

Before end users can access the application, you must deploy the application to a Data Integration Service. Or, an administrator must deploy the application to a Data Integration Service through the Administrator tool.
Updating an Application

Update an application when you want to add objects to an application, remove objects from an application, or update mapping deployment properties.

1. Open the application you want to update.
2. To add or remove objects, click the Overview view.
3. To add objects to the application, click Add.
   The Developer tool prompts you to choose the objects to add to the application.
4. To remove an object from the application, select the object, and click Remove.
5. To update mapping deployment properties, click the Advanced view and change the properties.
6. Save the application.
Redeploy the application if you want end users to be able to access the updated application.

Importing Application Archives

You can import objects from an application archive file. You import the application and dependent objects into the repository.

1. Click File > Import.
   The Import wizard appears.
2. Select Informatica > Application Archive.
3. Click Next.
4. Click Browse to select the application archive file.
   The Developer tool lists the application archive file contents.
5. Select the repository into which you want to import the application.
6. Click Finish.
   The Developer tool imports the application into the repository. If the Developer tool finds duplicate objects, it renames the imported objects.

Application Redeployment

When you change an application or change an object in the application and you want end users to access the latest version of the application, you must deploy the application again.

When you change an application or its contents and you deploy the application to the same Data Integration Service, the Developer tool gives you the following choices:

- Update. The Data Integration Service replaces the objects and preserves the object properties in the Administrator tool.
- Replace. The Data Integration Service replaces the objects and resets the object properties in the Administrator tool to the default values.
To update or replace an application that is running, you must first stop the application. When you stop an application, the Data Integration Service aborts all running objects in the application. If you do not want to abort running objects, you can rename the application or deploy the application to a different service.

When you change an application and deploy it to a network file system, the Developer tool allows you to replace the application archive file or cancel the deployment. If you replace the application archive file, the Developer tool replaces the objects in the application and resets the object properties.

**Redeploying an Application**

Redeploy an application to a Data Integration Service when you want to update or replace the application.

1. Right-click an application in the **Object Explorer** view and click **Deploy**.
   
The **Deploy** dialog box appears.

2. Select **Deploy to Service**.

3. Click **Browse** to select the domain.
   
The **Choose Domain** dialog box appears.

4. Select a domain and click **OK**.
   
The Developer tool lists the Data Integration Service associated with the domain in the **Available Services** section of the **Deploy Application** dialog box.

5. Select the Data Integration Service to which you want to deploy the application.

6. If the Data Integration Service already contains the deployed application, select to update or replace the application in the **Action** column.

7. If the deployed application is running, select **Force the Existing Application to Stop**.

8. Click **Finish**.
CHAPTER 14

Mapping Parameters and Parameter Files

This chapter includes the following topics:
- Mapping Parameters and Parameter Files Overview, 164
- System Parameters, 164
- User-Defined Parameters, 165
- Where to Assign Parameters, 167
- Parameter Files, 168

Mapping Parameters and Parameter Files Overview

A mapping parameter represents a constant value that can change between mapping runs, such as connections, source file directories, or cache file directories.

You can use system or user-defined parameters when you run a mapping. System parameters define the directories where the Data Integration Service stores cache files, reject files, source files, target files, and temporary files. You define the values of the system parameters on a Data Integration Service process in the Administrator tool.

User-defined parameters allow you to define mapping values in a parameter file and update those values each time you run a mapping. Create user-defined parameters so that you can rerun a mapping with different connection, flat file, cache file, or temporary file values. You define the parameter values in a parameter file. When you run the mapping from the command line and specify a parameter file, the Data Integration Service uses the parameter values defined in the parameter file.

Note: You can create user-defined workflow parameters when you develop a workflow. A workflow parameter is a constant value that can change between workflow runs.

System Parameters

System parameters are constant values that define the directories where the Data Integration Service stores cache files, reject files, source files, target files, and temporary files.

You define the values of the system parameters on a Data Integration Service process in the Administrator tool. You cannot define or override system parameter values in a parameter file.
You cannot create system parameters. The Developer tool provides a pre-defined list of system parameters that you can assign to a data object or transformation in a mapping. By default, the system parameters are assigned to flat file directory, cache file directory, and temporary file directory fields. For example, when you create an Aggregator transformation, the cache directory system parameter is the default value assigned to the cache directory field.

The following table describes the system parameters:

<table>
<thead>
<tr>
<th>System Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheDir</td>
<td>String</td>
<td>Default directory for index and data cache files.</td>
</tr>
<tr>
<td>RejectDir</td>
<td>String</td>
<td>Default directory for reject files.</td>
</tr>
<tr>
<td>SourceDir</td>
<td>String</td>
<td>Default directory for source files.</td>
</tr>
<tr>
<td>TargetDir</td>
<td>String</td>
<td>Default directory for target files.</td>
</tr>
<tr>
<td>TempDir</td>
<td>String</td>
<td>Default directory for temporary files.</td>
</tr>
</tbody>
</table>

User-Defined Parameters

User-defined parameters represent values that change between mapping runs. You can create user-defined parameters that represent connections, long values, or string values.

Create parameters so that you can rerun a mapping with different values. For example, you create a mapping that processes customer orders. The mapping reads customer information from a relational table that contains customer data for one country. You want to use the mapping for customers in the United States, Canada, and Mexico. Create a user-defined parameter that represents the connection to the customers table. Create three parameter files that set the connection name to the U.S. customers table, the Canadian customers table, and the Mexican customers table. Run the mapping from the command line, using a different parameter file for each mapping run.

You can create the following types of user-defined parameters:

- **Connection.** Represents a database connection. You cannot create connection parameters for enterprise application or social media connections.
- **Long.** Represents a long or integer value.
- **String.** Represents a flat file name, flat file directory, cache file directory, temporary file directory, or type of mapping run-time environment.

Process to Run Mappings with User-Defined Parameters

A user-defined parameter represents a constant value that you define in a parameter file before running a mapping.

To run mappings with different parameter values, perform the following tasks:

1. Create a user-defined parameter and assign it a default value.
2. Apply the parameter to the mapping or to a data object or transformation in the mapping.
3. Add the mapping to an application and deploy the application.
4. Create a parameter file that contains the user-defined parameter value.
5. Run the mapping from the command line with the parameter file.
Where to Create User-Defined Parameters

You can create user-defined parameters in physical data objects, some reusable transformations, mappings, and mapplets.

When you create a parameter in a physical data object or reusable transformation, you can use the parameter in the data object or transformation. When you create a parameter in a mapping or mapplet, you can use the parameter in any nonreusable data object, nonreusable transformation, or reusable Lookup transformation in the mapping or mapplet that accepts parameters. When you create a parameter in a mapping, you can also use the parameter in the mapping.

The following table lists the objects in which you can create user-defined parameters:

<table>
<thead>
<tr>
<th>Object</th>
<th>Parameter Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator transformation</td>
<td>String</td>
</tr>
<tr>
<td>Customized data object (reusable)</td>
<td>Connection</td>
</tr>
<tr>
<td>Flat file data object</td>
<td>Connection, String</td>
</tr>
<tr>
<td>Joiner transformation</td>
<td>String</td>
</tr>
<tr>
<td>Lookup transformation (relational lookups)</td>
<td>Connection</td>
</tr>
<tr>
<td>Mapping</td>
<td>Connection, Long, String</td>
</tr>
<tr>
<td>Mapplet</td>
<td>Connection, Long, String</td>
</tr>
<tr>
<td>Rank transformation</td>
<td>String</td>
</tr>
<tr>
<td>Sorter transformation</td>
<td>String</td>
</tr>
</tbody>
</table>

Creating a User-Defined Parameter

Create a user-defined parameter to represent a value that changes between mapping runs.

1. Open the physical data object, mapping, mapplet, or reusable transformation where you want to create a user-defined parameter.
2. Click the Parameters view.
3. Click Add.
   The Add Parameter dialog box appears.
4. Enter the parameter name.
5. Optionally, enter a parameter description.
6. Select the parameter type.
7. Enter a default value for the parameter.
   For connection parameters, select a connection. For other parameter types, enter a value.
8. Click OK.
   The Developer tool adds the parameter to the list of parameters.

Where to Assign Parameters

Assign a system parameter to a field when you want the Data Integration Service to replace the parameter with the value defined for the Data Integration Service process. Assign a user-defined parameter to a field when you want the Data Integration Service to replace the parameter with the value defined in a parameter file.

The following table lists the objects and fields where you can assign system or user-defined parameters:

<table>
<thead>
<tr>
<th>Object</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator transformation</td>
<td>Cache directory</td>
</tr>
<tr>
<td>Customized data object</td>
<td>Connection</td>
</tr>
<tr>
<td>Flat file data object</td>
<td>Source file name</td>
</tr>
<tr>
<td></td>
<td>Output file name</td>
</tr>
<tr>
<td></td>
<td>Source file directory</td>
</tr>
<tr>
<td></td>
<td>Output file directory</td>
</tr>
<tr>
<td></td>
<td>Connection name</td>
</tr>
<tr>
<td></td>
<td>Reject file directory</td>
</tr>
<tr>
<td>Joiner transformation</td>
<td>Cache directory</td>
</tr>
<tr>
<td>Lookup transformation (flat file lookups)</td>
<td>Lookup cache directory name</td>
</tr>
<tr>
<td>Lookup transformation (relational lookups)</td>
<td>Connection</td>
</tr>
<tr>
<td></td>
<td>Lookup cache directory name</td>
</tr>
<tr>
<td>Mapping</td>
<td>Run-time environment</td>
</tr>
<tr>
<td>Rank transformation</td>
<td>Cache directory</td>
</tr>
<tr>
<td>Read transformation created from related relational data objects</td>
<td>Connection</td>
</tr>
<tr>
<td>Sorter transformation</td>
<td>Work directory</td>
</tr>
</tbody>
</table>

Assigning a Parameter

Assign a system parameter to a field so that the Data Integration Service replaces the parameter with the value defined for the Data Integration Service process. Assign a user-defined parameter to a field so that when you run a
mapping from the command line, the Data Integration Service replaces the parameter with the value defined in the parameter file.

1. Open the field in which you want to assign a parameter.
2. Click **Assign Parameter**.
   The Assign Parameter dialog box appears.
3. Select the system or user-defined parameter.
4. Click **OK**.

### Parameter Files

A parameter file is an XML file that lists user-defined parameters and their assigned values. Parameter files provide the flexibility to change parameter values each time you run a mapping.

The parameter values define properties for a mapping, mapplet, physical data object, or transformation. The Data Integration Service applies these values when you run a mapping from the command line and specify a parameter file.

You cannot define system parameter values in a parameter file.

You can define parameters for multiple mappings in a single parameter file. You can also create multiple parameter files and use a different file each time you run a mapping. The Data Integration Service reads the parameter file at the start of the mapping run to resolve the parameters.

Use the `infacmd ms ListMappingParams` command to list the parameters used in a mapping with the default values. You can use the output of this command as a parameter file template.

Use the `infacmd ms RunMapping` command to run a mapping with a parameter file.

**Note:** Parameter files for mappings and workflows use the same structure. You can define parameters for deployed mappings and for deployed workflows in a single parameter file.

### Parameter File Structure

A parameter file is an XML file that contains at least one parameter and its assigned value.

The Data Integration Service uses the hierarchy defined in the parameter file to identify parameters and their defined values. The hierarchy identifies the mapping, mapplet, physical data object, or transformation that uses the parameter.

You define parameter values within a project or application top-level element. A project element defines parameter values to use when you run a specific mapping in the project in any deployed application. A project element also defines the parameter values to use when you run any mapping that uses the objects in the project. An application element defines parameter values to use when you run a specific mapping in a specific deployed application. If you define the same parameter in a project top-level element and an application top-level element in the same parameter file, the parameter value defined in the application element takes precedence.

The Data Integration Service searches for parameter values in the following order:

1. The value specified within an application element.
2. The value specified within a project element.
3. The parameter default value.
A parameter file must conform to the structure of the parameter file XML schema definition (XSD). If the parameter file does not conform to the schema definition, the Data Integration Service fails the mapping run.

On the machine that hosts the Developer tool, the parameter file XML schema definition appears in the following directory:

```
<Informatica Installation Directory>\clients\DeveloperClient\infacmd\plugins\ms
\parameter_file_schema_1_0.xsd
```

On the machine that hosts Informatica Services, the parameter file XML schema definition appears in the following directory:

```
<Informatica Installation Directory>\isp\bin\plugins\ms\parameter_file_schema_1_0.xsd
```

**Project Element**

A project element defines the parameter values to use when you run a specific mapping in the project in any deployed application. A project element also defines the parameter values to use when you run any mapping that uses the objects in the project.

The project element defines the project in the Model repository that contains objects that use parameters. The project element contains additional elements that define specific objects within the project.

The following table describes the elements that a project element can contain:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder</td>
<td>Defines a folder within the project. Use a folder element if objects are organized in multiple folders within the project. A folder element can contain a dataSource, mapping, mapplet, or transformation element.</td>
</tr>
<tr>
<td>dataSource</td>
<td>Defines a physical data object within the project that uses parameters. A dataSource element contains one or more parameter elements that define parameter values for the data object.</td>
</tr>
<tr>
<td>mapping</td>
<td>Defines a mapping within the project that uses parameters. A mapping element contains one or more parameter elements that define parameter values for the mapping or for any non-reusable data object, non-reusable transformation, or reusable Lookup transformation in the mapping that accepts parameters.</td>
</tr>
<tr>
<td>mapplet</td>
<td>Defines a mapplet within the project that uses parameters. A mapplet element contains one or more parameter elements that define parameter values for any non-reusable data object, non-reusable transformation, or reusable Lookup transformation in the mapplet that accepts parameters.</td>
</tr>
<tr>
<td>transformation</td>
<td>Defines a reusable transformation within the project that uses parameters. A transformation element contains one or more parameter elements that define parameter values for the transformation.</td>
</tr>
</tbody>
</table>

When you run a mapping with a parameter file that defines parameter values in a project top-level element, the Data Integration Service applies the parameter values to the specified mapping. The service also applies parameter values to any of the specified objects included in the mapping.

For example, you want the Data Integration Service to apply parameter values when you run mapping "MyMapping". The mapping includes data object "MyDataObject" and reusable transformation "MyTransformation". You want to use the parameter values when you run "MyMapping" in any deployed application. You also want to use the parameter values when you run any other mapping that uses "MyDataObject" and "MyTransformation" in project "MyProject". Define the parameter within the following elements:

```
<project name="MyProject">
  <!-- Apply this parameter value to mapping "MyMapping" in project "MyProject". -->
  <mapping name="MyMapping">
    <parameter name="MyMapping_Param">Param_value</parameter>
  </mapping>
</projec
```
Rules and Guidelines for Parameter Files

Certain rules and guidelines apply when you create parameter files.

Use the following rules when you create a parameter file:

- Parameter values cannot be empty. For example, the Data Integration Service fails the mapping run if the parameter file contains the following entry:

  ```xml
  <param name="Param1" />
  ```

- Within an element, artifact names are not case-sensitive. Therefore, the Data Integration Service interprets `<application name="App1">` and `<application name="APP1">` as the same application.

Sample Parameter File

The following example shows a sample parameter file used to run mappings.
<?xml version="1.0"?>
<root description="Sample Parameter File"
xmlns="http://www.informatica.com/Parameterization/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!--
The Data Integration Service uses this section only when you run mapping "Map1" or "Map2"
in project "Project1" in deployed application "App1."

This section assigns values to parameters created in mappings "Map1" and "Map2."
-->
<application name="App1">
  <mapping name="Map1">
    <project name="Project1">
      <parameter name="MAP1_PARAM1">MAP1_PARAM1_VAL</parameter>
      <parameter name="MAP1_PARAM2">MAP1_PARAM2_VAL</parameter>
    </project>
  </mapping>
</application>

<!--
The Data Integration Service uses this section only when you run mapping "Map1" in
project "Project2" in deployed application "App2."

This section assigns values to parameters created in the following objects:
* Reusable data source "DS1" in mapping "Map1"
* Mapping "Map1"
-->
<application name="App2">
  <mapping name="Map1">
    <project name="Project1">
      <dataSource name="DS1">
        <parameter name="PROJ1_DS1_APP2_MAP1_VAL">PROJ1_DS1_APP2_MAP1_VAL</parameter>
      </dataSource>
      <parameter name="Map1">MAP1_PARAM2_VAL</parameter>
    </project>
  </mapping>
</application>

<!--
The Data Integration Service uses this section when you run any mapping that
includes data source "DS1" or maplet "DS1" in project "Project1."

This section assigns values to parameters created in the following objects:
* Data source "DS1"
* Maplet "DS1"
-->
<project name="Project1">
  <dataSource name="DS1">
    <parameter name="PROJ1_DS1">PROJ1_DS1_VAL</parameter>
    <parameter name="PROJ1_DS1_PARAM1">PROJ1_DS1_PARAM1_VAL</parameter>
  </dataSource>
  <maplet name="DS1">
    <parameter name="PROJ1_DS1">PROJ1_DS1_VAL</parameter>
    <parameter name="PROJ1_DS1_PARAM1">PROJ1_DS1_PARAM1_VAL</parameter>
  </maplet>
</project>

<!--
The Data Integration Service uses this section when you run any mapping that
includes reusable transformation "TX2", maplet "MPLT1" in folder "Folder2",
or Maplet "RULE1" in nested folder "Folder2.I" in project "Project2."

This section assigns values to parameters created in the following objects:
Creating a Parameter File

The infacmd ms ListMappingParams command lists the parameters used in a mapping in a deployed application and the default value for each parameter. Use the output of this command to create a parameter file.

The command lists all parameters in a project top-level element. You can edit the parameter default values in the project element to define the values for a mapping in the project that is deployed to any application. Or, you can copy the project element into an application element to define the values for a specific mapping in a specific deployed application.

If the mapping uses objects of the same type that exist in the same project or folder, have the same name, and use parameters, the ms ListMappingParams command fails. If the objects are in different folders, or if one object does not use parameters, the ms ListMappingParams command successfully lists the parameters used in the mapping.

1. Run the infacmd ms ListMappingParams command to list all parameters used in a mapping and the default value for each parameter.
   The -o argument sends command output to an XML file.
   For example, the following command lists the parameters in mapping MyMapping in file "MyOutputFile.xml":
   ```bash
   infacmd ms ListMappingParams -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a MyApplication -m MyMapping -o MyOutputFile.xml
   ```
   The Data Integration Service lists all parameters in the mapping with their default values in a project top-level element.

2. If you did not specify the -o argument, copy the command output to an XML file and save the file.
3. Edit the XML file and replace the parameter default values with the values you want to use when you run the mapping.
   If you want to define the values for the mapping in a specific application, copy the project top-level element into an application top-level element.
4. Save the XML file.
Running a Mapping with a Parameter File

Use the infacmd ms RunMapping command to run a mapping with a parameter file. The -pf argument specifies the parameter file name.

For example, the following command runs the mapping MyMapping using the parameter file "MyParamFile.xml":

```bash
```

The Data Integration Service fails the mapping if you run it with a parameter file and any of the following conditions are true:

- The machine from which you run the infacmd ms RunMapping command cannot access the parameter file.
- The parameter file is not valid or does not exist.
- Objects of the same type exist in the same project or folder, have the same name, and use parameters.
CHAPTER 15

Object Import and Export

This chapter includes the following topics:

- Object Import and Export Overview, 174
- Import and Export Objects, 174
- Object Export, 175
- Object Import, 176

Object Import and Export Overview

You can export multiple objects from a project to one XML file. When you import objects, you can choose individual objects in the XML file or all the objects in the XML file.

You can export objects to an XML file and then import objects from the XML file. When you export objects, the Developer tool creates an XML file that contains the metadata of the exported objects. Use the XML file to import the objects into a project or folder. You can also import and export objects through infacmd command.

Export and import objects to accomplish the following tasks:

Deploy metadata into production

After you test a mapping in a development repository, you can export it to an XML file and then import it from the XML file into a production repository.

Archive metadata

You can export objects to an XML file that you no longer need before you remove them from the repository.

Share metadata

You can share metadata with a third party. For example, you can send a mapping to someone else for testing or analysis.

You can use infacmd to generate a readable XML file from an export file. You can also edit the object names in the readable XML and update the export XML before you import the objects into a repository.

Import and Export Objects

You can import and export projects and objects in a project. You can also import and export application archive files in a repository.
When you export an object, the Developer tool also exports the dependent objects. A dependent object is an object that is used by another object. For example, a physical data object used as a mapping input is a dependent object of that mapping. When you import an object, the Developer tool imports all the dependent objects.

When you export or import objects in a project or folder, the Model Repository Service preserves the object hierarchy.

The following table lists objects and dependent objects that you can export:

<table>
<thead>
<tr>
<th>Object</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>- Mappings, or workflows and their dependent objects</td>
</tr>
<tr>
<td>Project</td>
<td>- Projects contain other objects, but they do not have dependent objects</td>
</tr>
<tr>
<td>Folder</td>
<td>- Folders contain other objects, but they do not have dependent objects</td>
</tr>
<tr>
<td>Physical data object (except for customized data object)</td>
<td>- Physical data objects do not have dependent objects</td>
</tr>
<tr>
<td>Customized data object</td>
<td>- Physical data objects</td>
</tr>
<tr>
<td>Logical data object model</td>
<td>- Logical data objects</td>
</tr>
<tr>
<td></td>
<td>- Physical data objects</td>
</tr>
<tr>
<td></td>
<td>- Reusable transformations and their dependent objects</td>
</tr>
<tr>
<td></td>
<td>- Mapplets and their dependent objects</td>
</tr>
<tr>
<td>Transformation</td>
<td>- Physical data objects</td>
</tr>
<tr>
<td>Mapplet</td>
<td>- Logical data objects</td>
</tr>
<tr>
<td></td>
<td>- Physical data objects</td>
</tr>
<tr>
<td></td>
<td>- Reusable transformations and their dependent objects</td>
</tr>
<tr>
<td></td>
<td>- Mapplets and their dependent objects</td>
</tr>
<tr>
<td>Mapping</td>
<td>- Logical data objects</td>
</tr>
<tr>
<td></td>
<td>- Physical data objects</td>
</tr>
<tr>
<td></td>
<td>- Reusable transformations and their dependent objects</td>
</tr>
<tr>
<td></td>
<td>- Mapplets and their dependent objects</td>
</tr>
<tr>
<td>Profile</td>
<td>- Logical data objects</td>
</tr>
<tr>
<td></td>
<td>- Physical data objects</td>
</tr>
<tr>
<td>Workflow</td>
<td>- Mappings and their dependent objects</td>
</tr>
</tbody>
</table>

**Object Export**

When you export an object, the Developer tool creates an XML file that contains the metadata of the objects.

You can choose the objects to export. You must also choose to export all dependent objects. The Developer tool exports the objects and the dependent objects. The Developer tool exports the last saved version of the object. The Developer tool includes Cyclic Redundancy Checking Value (CRCVALUE) codes in the elements in the XML file. If you modify attributes in an element that contains a CRCVALUE code, you cannot import the object.

You can also export objects with the infacmd oie ExportObjects command.
**Exporting Objects**

You can export objects to an XML file to use in another project or folder.

1. Click **File > Export**.
2. Select **Informatica > Export Object Metadata File**.
3. Click **Next**.
4. Click **Browse** to select a project from which to export objects.
5. Click **Next**.
6. Select the objects to export.
7. Enter the export file name and location.
8. To view the dependent objects that the **Export** wizard exports with the objects you selected, click **Next**.
   - The **Export** wizard displays the dependent objects.
9. Click **Finish**.
   - The Developer tool exports the objects to the XML file.

**Object Import**

You can import a project or objects within project from an export file. You can import the objects and any dependent objects into a project or folder.

When you import objects, you can import a project or individual objects. Import a project when you want to reuse all objects in the project. Import individual objects when you want to reuse objects across projects. You cannot import objects from an export file that you created in a previous version.

When you import an object, the Developer Tool lists all the dependent objects. You must add each dependent object to the target before you can import the object.

When you import objects, an object in the export file might have the same name as an object in the target project or folder. You can choose how you want to resolve naming conflicts.

You can also import objects with the `infacmd oie ImportObjects` command.

**Importing Projects**

You can import a project from an XML file into the target repository. You can also import the contents of the project into a project in the target repository.

1. Click **File > Import**.
2. Select **Informatica > Import Object Metadata File (Basic)**.
3. Click **Next**.
4. Click **Browse** and select the export file that you want to import.
5. Click **Next**.
6. Select the project or select "<project name> Project Content" in the Source pane.
   - If you select the project in the Source pane, select the Model Repository Service in the Target pane where you want to import the project.
If you select the project content in the Source pane, select the project to which you want to import the project contents in the Target pane.

7. Click **Add to Target** to add the project to the target.
   
   **Tip:** You can also drag the project from the Source pane into the repository in the Target pane. Or, you can drag the project content in the Source pane into a project in the Target pane.

8. Click **Resolution** to specify how to handle duplicate objects.
   
   You can rename the imported object, replace the existing object with the imported object, or reuse the existing object. The Developer tool renames all the duplicate objects by default.

9. Click **Next**.
   
   The Developer tool summarizes the objects to be imported. Click **Link Source and Target Objects** to link the objects in the Source and Target display panes when you select one of the objects. For example, if you select this option and then select an object in the Source pane, the Developer tool selects the same object in the Target pane.

10. Map the connections from the import file to the target domain connections in the Additional Import Settings pane.
    You can also select whether to overwrite existing tags on the objects.

11. Click **Finish**.
    
    If you chose to rename the duplicate project, the Model Repository Service appends a number to the object name. You can rename the project after you import it.

**Importing Objects**

You can import objects from an XML file or application archive file. You import the objects and any dependent objects into a project.
1. Click **File > Import**.
2. Select **Informatica > Import Object Metadata File (Advanced)**.
3. Click **Next**.
4. Click **Browse** to select the export file that you want to import.
5. Click **Next**.
6. Select the object in the Source pane that you want to import.
7. Select the project in the Target pane to which you want to import the object.
8. Click **Add to Target** to add the object to the target.

   If you click **Auto Match to Target**, the Developer tool tries to match the descendants of the current source selection individually by name, type, and parent hierarchy in the target selection and adds the objects that match.

   If you want to import all the objects under a folder or a project, select the target folder or project and click **Add Content to Target**.

   **Tip:** You can also drag the object from the Source pane into the required project in the Target pane. Press the control key while you drag to maintain the object hierarchy in source and target.

9. Click to specify how to handle duplicate objects.

   You can rename the imported object, replace the existing object with the imported object, or reuse the existing object. The Developer tool renames all the duplicate objects by default.

10. Click **Next**.

    The Developer tool lists any dependent objects in the import file.

11. Add dependent objects to a target folder or project.
12. Click **Next**.

The Developer tool summarizes the objects to be imported. Click **Link Source and Target Objects** to link the objects in the Source and Target display panes when you select one of the objects. For example, if you select this option and then select an object in the Source pane, the Developer tool selects the same object in the Target pane.

13. Map the connections from the import file to the target domain connections in the Additional Import Settings pane. You can also select whether to overwrite existing tags on the objects.

14. Click **Finish**.

If you choose to rename the duplicate project, the Import wizard names the imported project as "<Original Name>_<number of the copy>." You can rename the project after you import it.
Chapter 16

Performance Tuning

This chapter includes the following topics:

- Optimizer Levels, 180
- Optimization Methods Overview, 181
- Full Optimization and Memory Allocation, 183
- Setting the Optimizer Level for a Developer Tool Mapping, 184
- Setting the Optimizer Level for a Deployed Mapping, 184

Optimizer Levels

The Data Integration Service attempts to apply different optimizer methods based on the optimizer level that you configure for the object.

You can choose one of the following optimizer levels:

- **None**
  - The Data Integration Service does not apply any optimization.

- **Minimal**
  - The Data Integration Service applies the early projection optimization method.

- **Normal**
  - The Data Integration Service applies the early projection, early selection, push-into, pushdown, and predicate optimization methods. Normal is the default optimization level.

- **Full**
  - The Data Integration Service applies the cost-based, early projection, early selection, predicate, push-into, pushdown, and semi-join optimization methods.
Optimization Methods Overview

The Data Integration Service applies optimization methods to reduce the number of rows to process in the mapping. You can configure the optimizer level for the mapping to limit which optimization methods the Data Integration Service applies.

The Data Integration Service can apply the following optimization methods:

- Pushdown optimization
- Early projection
- Early selection
- Push-into optimization
- Predicate optimization
- Cost-based
- Semi-join

The Data Integration Service can apply multiple optimization methods to a mapping at the same time. For example, the Data Integration Service applies the early projection, predicate optimization, and early selection or push-into optimization methods when you select the normal optimizer level.

Early Projection Optimization Method

When the Data Integration Service applies the early projection optimization method, it identifies unused ports and removes the links between those ports.

Early projection improves performance by reducing the amount of data that the Data Integration Service moves across transformations. When the Data Integration Service processes a mapping, it moves the data from all connected ports in a mapping from one transformation to another. In large, complex mappings, or in mappings that use nested mapplets, some ports might not supply data to the target. The Data Integration Service identifies the ports that do not supply data to the target. After the Data Integration Service identifies unused ports, it removes the links between all unused ports from the mapping.

The Data Integration Service does not remove all links. For example, it does not remove the following links:

- Links connected to a Custom transformation
- Links connected to transformations that call an ABORT() or ERROR() function, send email, or call a stored procedure

If the Data Integration Service determines that all ports in a transformation are unused, it removes all transformation links except the link to the port with the least data. The Data Integration Service does not remove the unused transformation from the mapping.

The Developer tool enables this optimization method by default.

Early Selection Optimization Method

When the Data Integration Service applies the early selection optimization method, it splits, moves, or removes the Filter transformations in a mapping. It moves filters up the mapping closer to the source.
The Data Integration Service might split a Filter transformation if the filter condition is a conjunction. For example, the Data Integration Service might split the filter condition "A>100 AND B<50" into two simpler conditions, "A>100" and "B<50." When the Data Integration Service splits a filter, it moves the simplified filters up the mapping pipeline, closer to the source. The Data Integration Service moves the filters up the pipeline separately when it splits the filter.

The Developer tool enables the early selection optimization method by default when you choose a normal or full optimizer level. The Data Integration Service does not enable early selection if a transformation that appears before the Filter transformation has side effects. You can configure the SQL transformation, Web Service Consumer transformation, and Java transformation for early selection optimization, however the Developer tool cannot determine if the transformations have side effects.

You can disable early selection if the optimization does not increase performance.

**Predicate Optimization Method**

When the Data Integration Service applies the predicate optimization method, it examines the predicate expressions that a mapping generates. It determines whether it can simplify or rewrite the expressions to increase mapping performance.

When the Data Integration Service runs a mapping, it generates queries against the mapping sources and performs operations on the query results based on the mapping logic and the transformations within the mapping. The queries and operations often include predicate expressions. Predicate expressions represent the conditions that the data must satisfy. The filter and join conditions in Filter and Joiner transformations are examples of predicate expressions.

With the predicate optimization method, the Data Integration Service also attempts to apply predicate expressions as early as possible in the mapping to improve mapping performance.

The Data Integration Service infers relationships from by existing predicate expressions and creates new predicate expressions. For example, a mapping contains a Joiner transformation with the join condition "A=B" and a Filter transformation with the filter condition "A>5." The Data Integration Service might be able to add "B>5" to the join condition.

The Data Integration Service applies the predicate optimization method with the early selection optimization method when it can apply both methods to a mapping. For example, when the Data Integration Service creates new filter conditions through the predicate optimization method, it also attempts to move them upstream in the mapping through the early selection method. Applying both optimization methods improves mapping performance when compared to applying either method alone.

The Data Integration Service applies the predicate optimization method if the application increases performance. The Data Integration Service does not apply this method if the application changes the mapping results or it decreases the mapping performance.

**Cost-Based Optimization Method**

With cost-based optimization, the Data Integration Service evaluates a mapping, generates semantically equivalent mappings, and runs the mapping with the best performance. Cost-based optimization reduces run time for mappings that perform adjacent, unsorted, inner-join operations.

Semantically equivalent mappings are mappings that perform identical functions and produce the same results. To generate semantically equivalent mappings, the Data Integration Service divides the original mapping into fragments. The Data Integration Service then determines which mapping fragments it can optimize.

The Data Integration Service optimizes each fragment that it can optimize. During optimization, the Data Integration Service might add, remove, or reorder transformations within a fragment. The Data Integration Service verifies that the optimized fragments produce the same results as the original fragments and forms alternate mappings that use the optimized fragments.
The Data Integration Service generates all or almost all of the mappings that are semantically equivalent to the original mapping. It uses profiling or database statistics to compute the cost for the original mapping and each alternate mapping. Then, it identifies the mapping that runs most quickly. The Data Integration Service performs a validation check on the best alternate mapping to ensure that it is valid and produces the same results as the original mapping.

The Data Integration Service caches the best alternate mapping in memory. When you run a mapping, the Data Integration Service retrieves the alternate mapping and runs it instead of the original mapping.

**Semi-Join Optimization Method**

The semi-join optimization method attempts to reduce the amount of data extracted from the source by modifying join operations in the mapping.

The Data Integration Service applies this method to a Joiner transformation when one input group has many more rows than the other and when the larger group has many rows with no match in the smaller group based on the join condition. The Data Integration Service attempts to decrease the size of the data set of one join operand by reading the rows from the smaller group, finding the matching rows in the larger group, and then performing the join operation. Decreasing the size of the data set improves mapping performance because the Data Integration Service no longer reads unnecessary rows from the larger group source. The Data Integration Service moves the join condition to the larger group source and reads only the rows that match the smaller group.

Before applying this optimization method, the Data Integration Service performs analyses to determine whether semi-join optimization is possible and likely to be worthwhile. If the analyses determine that this method is likely to increase performance, the Data Integration Service applies it to the mapping. The Data Integration Service then reanalyzes the mapping to determine whether there are additional opportunities for semi-join optimization. It performs additional optimizations if appropriate. The Data Integration Service does not apply semi-join optimization unless the analyses determine that there is a high probability for improved performance.

The Developer tool does not enable this method by default.

**Full Optimization and Memory Allocation**

When you configure full optimization for a mapping, you might need to increase the available memory to avoid mapping failure.

When a mapping contains Joiner transformations and other transformations that use caching, the mapping might run successfully at the default optimization level. If you change the optimization level to full optimization and the Data Integration Service performs semi-join optimization, the Data Integration Service requires more memory to sort the data. The mapping might fail if you do not increase the maximum session size.

Change the **Maximum Session Size** in the **Execution Options** for the Data Integration Service process. Increase the **Maximum Session Size** by 50MB to 100MB.
Setting the Optimizer Level for a Developer Tool Mapping

When you run a mapping through the Run menu or mapping editor, the Developer tool runs the mapping with the normal optimizer level. To run the mapping with a different optimizer level, run the mapping from the Run Configurations dialog box.

1. Open the mapping.
2. Select Run > Open Run Dialog.
   The Run Configurations dialog box appears.
3. Select a mapping configuration that contains the optimizer level you want to apply or create a mapping configuration.
4. Click the Advanced tab.
5. Change the optimizer level, if necessary.
6. Click Apply.
7. Click Run to run the mapping.
   The Developer tool runs the mapping with the optimizer level in the selected mapping configuration.

Setting the Optimizer Level for a Deployed Mapping

Set the optimizer level for a mapping you run from the command line by changing the mapping deployment properties in the application.

The mapping must be in an application.

1. Open the application that contains the mapping.
2. Click the Advanced tab.
3. Select the optimizer level.
4. Save the application.

After you change the optimizer level, you must redeploy the application.
Pushdown Optimization

This chapter includes the following topics:
- Pushdown Optimization Overview, 185
- Pushdown Optimization to Sources, 186
- Pushdown Optimization Expressions, 187
- Comparing the Output of the Data Integration Service and Sources, 191

Pushdown Optimization Overview

Pushdown optimization causes the Data Integration Service to push transformation logic to the source database. The Data Integration Service translates the transformation logic into SQL queries and sends the SQL queries to the database. The source database executes the SQL queries to process the transformations.

Pushdown optimization improves the performance of mappings when the source database can process transformation logic faster than the Data Integration Service. The Data Integration Service also reads less data from the source.

The amount of transformation logic that the Data Integration Service pushes to the source database depends on the database, the transformation logic, and the mapping configuration. The Data Integration Service processes all transformation logic that it cannot push to a database.

The Data Integration Service can push the following transformation logic to the source database:
- Expression transformation logic
- Filter transformation logic
- Joiner transformation logic. The sources must be in the same database management system and must use identical connections.

The Data Integration Service cannot push transformation logic after a source in the following circumstances:
- The Data Integration Service cannot push any transformation logic if the source is a customized data object that contains a custom SQL query.
- The Data Integration Service cannot push any transformation logic if the source contains a column with a binary datatype.
- The Data Integration Service cannot push Expression or Joiner transformation logic if the source is a customized data object that contains a filter condition or user-defined join.

The Data Integration Service applies pushdown optimization to a mapping when you select the normal or full optimizer level. When you select the normal optimizer level, the Data Integration Service applies pushdown optimization after it
applies all other optimization methods. If you select the full optimizer level, the Data Integration Service applies pushdown optimization before semi-join optimization, but after all of the other optimization methods.

When you apply pushdown optimization, the Data Integration Service analyzes the optimized mapping from the source to the target or until it reaches a downstream transformation that it cannot push to the source database. The Data Integration Service generates and executes a SELECT statement based on the transformation logic for each transformation that it can push to the database. Then, it reads the results of this SQL query and processes the remaining transformations in the mapping.

**Related Topics:**
- “Optimizer Levels” on page 180

### Pushdown Optimization to Sources

The Data Integration Service can push transformation logic to different sources. The type of logic that the Data Integration Service pushes depends on the source type. The Data Integration Service can push Expression, Filter, and Joiner transformation logic to some sources. It can push Filter transformation logic to other sources.

The Data Integration Service can push transformation logic to the following types of sources:
- Sources that use native database drivers
- Sources that use ODBC drivers

### Pushdown Optimization to Native Sources

When the Data Integration Service pushes transformation logic to relational sources using the native drivers, the Data Integration Service generates SQL statements that use the native database SQL.

The Data Integration Service can push Expression, Filter, and Joiner transformation logic to the following native sources:
- IBM DB2 for Linux, UNIX, and Windows ("DB2 for LUW")
- Microsoft SQL Server
  The Data Integration Service can use a native connection to Microsoft SQL Server when the Data Integration Service runs on Windows.
- Oracle

### Pushdown Optimization to ODBC Sources

The Data Integration Service can push Expression, Filter, and Joiner transformation logic to databases that use ODBC drivers.

When you use ODBC to connect to a source, the Data Integration Service can generate SQL statements using ANSI SQL or native database SQL. The Data Integration Service can push more transformation logic to the source when it generates SQL statements using the native database SQL. The source can process native database SQL faster than it can process ANSI SQL.

You can specify the ODBC provider in the ODBC connection object. When the ODBC provider is database specific, the Data Integration Service can generates SQL statements using native database SQL. When the ODBC provider is **Other**, the Data Integration Service generates SQL statements using ANSI SQL.
You can configure a specific ODBC provider for the following ODBC connection types:

- Sybase ASE
- Microsoft SQL Server

Use an ODBC connection to connect to Microsoft SQL Server when the Data Integration Service runs on UNIX or Linux. Use a native connection to Microsoft SQL Server when the Data Integration Service runs on Windows.

## Pushdown Optimization Expressions

The Data Integration Service can push transformation logic to the source database when the transformation contains operators and functions that the source supports. The Data Integration Service translates the transformation expression into a query by determining equivalent operators and functions in the database. If there is no equivalent operator or function, the Data Integration Service processes the transformation logic.

If the source uses an ODBC connection and you configure a database-specific ODBC provider in the ODBC connection object, then the Data Integration Service considers the source to be the native source type.

### Functions

The following table summarizes the availability of Informatica functions for pushdown optimization. In each column, an X indicates that the Data Integration Service can push the function to the source.

<table>
<thead>
<tr>
<th>Function</th>
<th>DB2 for LUW</th>
<th>Microsoft SQL Server</th>
<th>ODBC</th>
<th>Oracle</th>
<th>Sybase ASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ADD_TO_DATE()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ASCII()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CEIL()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CHR()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CONCAT()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>COS()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>COSH()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DATE_COMPARE()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DECODE()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>EXP()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
</tr>
<tr>
<td>FLOOR()</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GET_DATE_PART()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>DB2 for LUW</td>
<td>Microsoft SQL Server</td>
<td>ODBC</td>
<td>Oracle</td>
<td>Sybase ASE</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>IIF()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
</tr>
<tr>
<td>IN()</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
</tr>
<tr>
<td>INITCAP()</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>INSTR()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ISNULL()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LAST_DAY()</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>LENGTH()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LN()</td>
<td>X</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LOG()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LOOKUP()</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>LOWER()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LPAD()</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>LTRIM()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MOD()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>POWER()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ROUND(DATE)</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>ROUND(NUMBER)</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RPAD()</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>RTRIM()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SIGN()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SIN()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SINH()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SOUNDEX()</td>
<td>X¹</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SQRT()</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SUBSTR()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYSDATE()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYSTIMESTAMP()</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### IBM DB2 Function Exceptions

The Data Integration Service cannot push supported functions to IBM DB2 for LUW sources under certain conditions.

The Data Integration Service processes transformation logic for IBM DB2 sources when expressions contain supported functions with the following logic:

- **ADD_TO_DATE** or **GET_DATE_PART** returns results with millisecond or nanosecond precision.
- **LTRIM** includes more than one argument.
- **RTRIM** includes more than one argument.
- **TO_BIGINT** converts a string to a bigint value on a DB2 for LUW source.
- **TO_CHAR** converts a date to a character string and specifies a format that is not supported by DB2.
- **TO_DATE** converts a character string to a date and specifies a format that is not supported by DB2.
- **TO_DECIMAL** converts a string to a decimal value without the scale argument.
- **TO_FLOAT** converts a string to a double-precision floating point number.
- **TO_INTEGER** converts a string to an integer value on a DB2 for LUW source.
Microsoft SQL Server Function Exceptions

The Data Integration Service cannot push supported functions to Microsoft SQL Server sources under certain conditions.

The Data Integration Service processes transformation logic for Microsoft SQL Server sources when expressions contain supported functions with the following logic:

- IN includes the CaseFlag argument.
- INSTR includes more than three arguments.
- LTRIM includes more than one argument.
- RTRIM includes more than one argument.
- TO_BIGINT includes more than one argument.
- TO_INTEGER includes more than one argument.

Oracle Function Exceptions

The Data Integration Service cannot push supported functions to Oracle sources under certain conditions.

The Data Integration Service processes transformation logic for Oracle sources when expressions contain supported functions with the following logic:

- ADD_TO_DATE or GET_DATE_PART returns results with subsecond precision.
- ROUND rounds values to seconds or subseconds.
- SYSTIMESTAMP returns the date and time with microsecond precision.
- TRUNC truncates seconds or subseconds.

ODBC Function Exception

The Data Integration Service processes transformation logic for ODBC when the CaseFlag argument for the IN function is a number other than zero.

Note: When the ODBC connection object properties include a database-specific ODBC provider, the Data Integration Service considers the source to be the native source type.

Sybase ASE Function Exceptions

The Data Integration Service cannot push supported functions to Sybase ASE sources under certain conditions.

The Data Integration Service processes transformation logic for Sybase ASE sources when expressions contain supported functions with the following logic:

- IN includes the CaseFlag argument.
- INSTR includes more than two arguments.
- LTRIM includes more than one argument.
- RTRIM includes more than one argument.
- TO_BIGINT includes more than one argument.
- TO_INTEGER includes more than one argument.
- TRUNC(Numbers) includes more than one argument.
Operators

The following table summarizes the availability of Informatica operators by source type. In each column, an X indicates that the Data Integration Service can push the operator to the source.

**Note:** Nonrelational sources are IMS, VSAM, and sequential data sets on z/OS.

<table>
<thead>
<tr>
<th>Operator</th>
<th>DB2 for LUW</th>
<th>Microsoft SQL Server</th>
<th>ODBC</th>
<th>Oracle</th>
<th>Sybase ASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>/</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>%</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>n/a</td>
</tr>
<tr>
<td>=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&lt;</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>!=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>^=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AND</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Comparing the Output of the Data Integration Service and Sources

The Data Integration Service and sources can produce different results when processing the same transformation logic. When the Data Integration Service pushes transformation logic to the source, the output of the transformation logic can be different.

**Case sensitivity**

The Data Integration Service and a database can treat case sensitivity differently. For example, the Data Integration Service uses case-sensitive queries and the database does not. A Filter transformation uses the following filter condition: IIF(col_varchar2 = 'CA', TRUE, FALSE). You need the database to return rows that match ‘CA.’ However, if you push this transformation logic to a database that is not case sensitive, it returns rows that match the values ‘Ca,’ ‘ca,’ ‘cA,’ and ‘CA.’
**Numeric values converted to character values**

The Data Integration Service and a database can convert the same numeric value to a character value in different formats. The database might convert numeric values to an unacceptable character format. For example, a table contains the number 1234567890. When the Data Integration Service converts the number to a character value, it inserts the characters ‘1234567890.’ However, a database might convert the number to ‘1.2E9.’ The two sets of characters represent the same value.

**Date formats for TO_CHAR and TO_DATE functions**

The Data Integration Service uses the date format in the TO_CHAR or TO_DATE function when the Data Integration Service pushes the function to the database. Use the TO_DATE functions to compare date or time values. When you use TO_CHAR to compare date or time values, the database can add a space or leading zero to values such as a single-digit month, single-digit day, or single-digit hour. The database comparison results can be different from the results of the Data Integration Service when the database adds a space or a leading zero.

**Precision**

The Data Integration Service and a database can have different precision for particular datatypes. Transformation datatypes use a default numeric precision that can vary from the native datatypes. The results can vary if the database uses a different precision than the Data Integration Service.

**SYSDATE or SYSTIMESTAMP function**

When you use the SYSDATE or SYSTIMESTAMP, the Data Integration Service returns the current date and time for the node that runs the service process. However, when you push the transformation logic to the database, the database returns the current date and time for the machine that hosts the database. If the time zone of the machine that hosts the database is not the same as the time zone of the machine that runs the Data Integration Service process, the results can vary.

If you push SYSTIMESTAMP to an IBM DB2 or a Sybase ASE database, and you specify the format for SYSTIMESTAMP, the database ignores the format and returns the complete time stamp.

**LTRIM, RTRIM, or SOUNDEX function**

When you push LTRIM, RTRIM, or SOUNDEX to a database, the database treats the argument (" ") as NULL, but the Data Integration Service treats the argument (" ") as spaces.

**LAST_DAY function on Oracle source**

When you push LAST_DAY to Oracle, Oracle returns the date up to the second. If the input date contains subseconds, Oracle trims the date to the second.
Datatype Reference

This appendix includes the following topics:
- Datatype Reference Overview, 193
- Transformation Datatypes, 194
- Flat File and Transformation Datatypes, 198
- IBM DB2 and Transformation Datatypes, 199
- JDBC and Transformation Datatypes, 200
- Microsoft SQL Server and Transformation Datatypes, 201
- ODBC and Transformation Datatypes, 203
- Oracle and Transformation Datatypes, 204
- XML and Transformation Datatypes, 206
- Converting Data, 207

Datatype Reference Overview

When you create a mapping, you create a set of instructions for the Data Integration Service to read data from a source, transform it, and write it to a target. The Data Integration Service transforms data based on dataflow in the mapping, starting at the first transformation in the mapping, and the datatype assigned to each port in a mapping.

The Developer tool displays two types of datatypes:
- Native datatypes. Specific to the relational table or flat file used as a physical data object. Native datatypes appear in the physical data object column properties.
- Transformation datatypes. Set of datatypes that appear in the transformations. They are internal datatypes based on ANSI SQL-92 generic datatypes, which the Data Integration Service uses to move data across platforms. The transformation datatypes appear in all transformations in a mapping.

When the Data Integration Service reads source data, it converts the native datatypes to the comparable transformation datatypes before transforming the data. When the Data Integration Service writes to a target, it converts the transformation datatypes to the comparable native datatypes.

When you specify a multibyte character set, the datatypes allocate additional space in the database to store characters of up to three bytes.
## Transformation Datatypes

The following table describes the transformation datatypes:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Size in Bytes</th>
<th>Description</th>
</tr>
</thead>
</table>
| Bigint     | 8 bytes       | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807  
Precision of 19, scale of 0  
Integer value. |
| Binary     | Precision     | 1 to 104,857,600 bytes  
You cannot use binary data for flat file sources. |
| Date/Time  | 16 bytes      | Jan 1, 0001 A.D. to Dec 31, 9999 A.D.  
Precision of 29, scale of 9  
(precision to the nanosecond)  
Combined date/time value. |
| Decimal    | 8 bytes (if high precision is off or precision is greater than 28)  
16 bytes (if precision <= 18 and high precision is on)  
20 bytes (if precision >18 and <= 28) | Precision 1 to 28 digits, scale 0 to 28  
Decimal value with declared precision and scale. Scale must be less than or equal to precision. |
| Double     | 8 bytes       | Precision of 15 digits  
Double-precision floating-point numeric value. |
| Integer    | 4 bytes       | -2,147,483,648 to 2,147,483,647  
Precision of 10, scale of 0  
Integer value. |
| String     | Unicode mode: (precision + 1) * 2  
ASCII mode: precision + 1 | 1 to 104,857,600 characters  
Fixed-length or varying-length string. |
| Text       | Unicode mode: (precision + 1) * 2  
ASCII mode: precision + 1 | 1 to 104,857,600 characters  
Fixed-length or varying-length string. |

### Integer Datatypes

You can pass integer data from sources to targets and perform transformations on integer data. The transformation language supports Bigint and Integer datatypes.

The transformation integer datatypes represent exact values.

### Integer Values in Calculations

When you use integer values in calculations, the Integration Service sometimes converts integer values to floating-point numbers before it performs the calculation. For example, to evaluate MOD( 12.00, 5 ), the Integration Service converts the integer value ”5” to a floating-point number before it performs the division operation. The Integration Service converts integer values to double or decimal values depending on whether you enable high precision.
The Integration Service converts integer values in the following arithmetic operations:

<table>
<thead>
<tr>
<th>Arithmetic Operation</th>
<th>High Precision Disabled</th>
<th>High Precision Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions and calculations that cannot introduce decimal points.</td>
<td>No conversion</td>
<td>Decimal</td>
</tr>
<tr>
<td>For example, integer addition, subtraction, and multiplication, and functions such as CUME, MOVINGSUM, and SUM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-scientific functions and calculations that can introduce decimal points.</td>
<td>Double</td>
<td>Decimal</td>
</tr>
<tr>
<td>For example, integer division, and functions such as AVG, MEDIAN, and PERCENTILE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All scientific functions and the EXP, LN, LOG, POWER, and SQRT functions.</td>
<td>Double</td>
<td>Double</td>
</tr>
</tbody>
</table>

1. If the calculation produces a result that is out of range, the Integration Service writes a row error.

The transformation Double datatype supports precision of up to 15 digits, while the Bigint datatype supports precision of up to 19 digits. Therefore, precision loss can occur in calculations that produce Bigint values with precision of more than 15 digits.

For example, an expression transformation contains the following calculation:

```python
POW(BIGNIVAL, EXPVAL)
```

Before it performs the calculation, the Integration Service converts the inputs to the POWER function to double values. If the BIGNIVAL port contains the Bigint value 9223372036854775807, the Integration Service converts this value to 9.22337203685478e+18, losing the last 4 digits of precision. If the EXPVAL port contains the value 1.0 and the result port is a Bigint, this calculation produces a row error since the result, 9223372036854780000, exceeds the maximum bigint value.

When you use an Integer datatype in a calculation that can produce decimal values and you enable high precision, the Integration Service converts the integer values to decimal values. The transformation Decimal datatype supports precision of up to 28 digits. Therefore, precision loss does not occur in a calculation unless the result produces a value with precision greater than 28 digits. In this case, the Integration Service stores the result as a double.

### Integer Constants in Expressions

The Integration Service interprets constants in an expression as floating-point values, even if the calculation produces an integer result. For example, in the expression INTVALUE + 1000, the Integration Service converts the integer value “1000” to a double value if high precision is not enabled. It converts the value 1000 to a decimal value if high precision is enabled. To process the value 1000 as an integer value, create a variable port with an Integer datatype to hold the constant and modify the expression to add the two ports.

### NaN Values

NaN (Not a Number) is a value that is usually returned as the result of an operation on invalid input operands, especially in floating-point calculations. For example, when an operation attempts to divide zero by zero, it returns a NaN result.

Operating systems and programming languages may represent NaN differently. For example the following list shows valid string representations of NaN:

- nan
- NaN
- NaNh
The Integration Service converts QNAN values to 1.#QNAN on Win64EMT platforms. 1.#QNAN is a valid representation of NaN.

Convert String Values to Integer Values

When the Integration Service performs implicit conversion of a string value to an integer value, the string must contain numeric characters only. Any non-numeric characters result in a transformation row error. For example, you link a string port that contains the value “9,000,000,000,000,000,000.777” to a Bigint port. The Integration Service cannot convert the string to a bigint value and returns an error.

Write Integer Values to Flat Files

When writing integer values to a fixed-width flat file, the file writer does not verify that the data is within range. For example, the file writer writes the result 3,000,000,000 to a target Integer column if the field width of the target column is at least 13. The file writer does not reject the row because the result is outside the valid range for Integer values.

Binary Datatype

If a mapping includes binary data, set the precision for the transformation binary datatype so that the Integration Service can allocate enough memory to move the data from source to target.

You cannot use binary datatypes for flat file sources.

Date/Time Datatype

The Date/Time datatype handles years from 1 A.D. to 9999 A.D. in the Gregorian calendar system. Years beyond 9999 A.D. cause an error.

The Date/Time datatype supports dates with precision to the nanosecond. The datatype has a precision of 29 and a scale of 9. Some native datatypes have a smaller precision. When you import a source that contains datetime values, the import process imports the correct precision from the source column. For example, the Microsoft SQL Server Datetime datatype has a precision of 23 and a scale of 3. When you import a Microsoft SQL Server source that contains Datetime values, the Datetime columns in the mapping source have a precision of 23 and a scale of 3.

The Integration Service reads datetime values from the source to the precision specified in the mapping source. When the Integration Service transforms the datetime values, it supports precision up to 29 digits. For example, if you import a datetime value with precision to the millisecond, you can use the ADD_TO_DATE function in an Expression transformation to add nanoseconds to the date.

If you write a Date/Time value to a target column that supports a smaller precision, the Integration Service truncates the value to the precision of the target column. If you write a Date/Time value to a target column that supports a larger precision, the Integration Service inserts zeroes in the unsupported portion of the datetime value.
Decimal and Double Datatypes

You can pass decimal and double data from sources to targets and perform transformations on decimal and double data. The transformation language supports the following datatypes:

- **Decimal.** Precision 1 to 28 digits, scale 0 to 28. You cannot use decimal values with scale greater than precision or a negative precision. Transformations display any range you assign to a Decimal datatype, but the Integration Service supports precision only up to 28.
- **Double.** Precision of 15.

Decimal and Double Values in Calculations

The transformation Decimal datatype supports precision of up to 28 digits and the Double datatype supports precision of up to 15 digits. Precision loss can occur with either datatype in a calculation when the result produces a value with a precision greater than the maximum.

If you disable high precision, the Integration Service converts decimal values to doubles. Precision loss occurs if the decimal value has a precision greater than 15 digits. For example, you have a mapping with Decimal (20,0) that passes the number 40012030304957666903. If you disable high precision, the Integration Service converts the decimal value to double and passes 4.00120303049577 x 10^{19}.

To ensure precision of up to 28 digits, use the Decimal datatype and enable high precision. When you enable high precision, the Integration Service processes decimal values as Decimal. Precision loss does not occur in a calculation unless the result produces a value with precision greater than 28 digits. In this case, the Integration Service stores the result as a double. Do not use the Double datatype for data that you use in an equality condition, such as a lookup or join condition.

The following table lists how the Integration Service handles decimal values based on high precision configuration:

<table>
<thead>
<tr>
<th>Port Datatype</th>
<th>Precision</th>
<th>High Precision Off</th>
<th>High Precision On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>0-28</td>
<td>Double</td>
<td>Decimal</td>
</tr>
<tr>
<td>Decimal</td>
<td>Over 28</td>
<td>Double</td>
<td>Double</td>
</tr>
</tbody>
</table>

When you enable high precision, the Integration Service converts numeric constants in any expression function to Decimal. If you do not enable high precision, the Integration Service converts numeric constants to Double.

To ensure the maximum precision for numeric values greater than 28 digits, truncate or round any large numbers before performing any calculations or transformations with the transformation functions.

Rounding Methods for Double Values

Due to differences in system run-time libraries and the computer system where the database processes double datatype calculations, the results may not be as expected. The double datatype conforms to the IEEE 794 standard. Changes to database client library, different versions of a database or changes to a system run-time library affect the binary representation of mathematically equivalent values. Also, many system run-time libraries implement the round-to-even or the symmetric arithmetic method. The round-to-even method states that if a number falls midway between the next higher or lower number it round to the nearest value with an even least significant bit. For example, with the round-to-even method, 0.125 is rounded to 0.12. The symmetric arithmetic method rounds the number to next higher digit when the last digit is 5 or greater. For example, with the symmetric arithmetic method 0.125 is rounded to 0.13 and 0.124 is rounded to 0.12.

To provide calculation results that are less susceptible to platform differences, the Integration Service stores the 15 significant digits of double datatype values. For example, if a calculation on Windows returns the number
1234567890.1234567890, and the same calculation on UNIX returns 1234567890.1234569999, the Integration Service converts this number to 1234567890.1234600000.

String Datatypes

The transformation datatypes include the following string datatypes:

- String
- Text

Although the String and Text datatypes support the same precision up to 104,857,600 characters, the Integration Service uses String to move string data from source to target and Text to move text data from source to target. Because some databases store text data differently than string data, the Integration Service needs to distinguish between the two types of character data. In general, the smaller string datatypes, such as Char and Varchar, display as String in transformations, while the larger text datatypes, such as Text, Long, and Long Varchar, display as Text.

Use String and Text interchangeably within transformations. However, in Lookup transformations, the target datatypes must match. The database drivers need to match the string datatypes with the transformation datatypes, so that the data passes accurately. For example, Varchar in a lookup table must match String in the Lookup transformation.

Flat File and Transformation Datatypes

Flat file datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The following table compares flat file datatypes to transformation datatypes:

<table>
<thead>
<tr>
<th>Flat File</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigint</td>
<td>Bigint</td>
<td>Precision of 19 digits, scale of 0</td>
</tr>
<tr>
<td>Datetime</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
<td>Precision of 15 digits</td>
</tr>
<tr>
<td>Int</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Nstring</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Number</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
</tbody>
</table>

When the Data Integration Service reads non-numeric data in a numeric column from a flat file, it drops the row and writes a message in the log. Also, when the Data Integration Service reads non-datetime data in a datetime column from a flat file, it drops the row and writes a message in the log.
IBM DB2 and Transformation Datatypes

IBM DB2 datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The following table compares IBM DB2 datatypes and transformation datatypes:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Range</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blob</td>
<td>1 to 2,147,483,647 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Char</td>
<td>1 to 254 characters</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Char for bit data</td>
<td>1 to 254 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Clob</td>
<td>1 to 2,447,483,647 bytes</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Date</td>
<td>0001 to 9999 A.D. (precision to the day)</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td>Decimal</td>
<td>Precision 1 to 31, scale 0 to 31</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Float</td>
<td>Precision 1 to 15</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Smallint</td>
<td>-32,768 to 32,767</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Time</td>
<td>24-hour time period (precision to the second)</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>26 bytes (precision to the microsecond)</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td>Varchar</td>
<td>Up to 4,000 characters</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Varchar for bit data</td>
<td>Up to 4,000 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
</tbody>
</table>

Unsupported IBM DB2 Datatypes

The Developer tool does not support certain IBM DB2 datatypes.

The Developer tool does not support the following IBM DB2 datatypes:

- Dbclob
- Graphic
JDBC and Transformation Datatypes

When the Data Integration Service reads data from a JDBC source, it converts the native datatypes into the corresponding JDBC datatypes and then to the transformation datatypes. It uses the transformation datatypes to move data across platforms.

The following table compares the JDBC datatypes to the transformation datatypes:

<table>
<thead>
<tr>
<th>JDBC Datatype</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Precision 19, scale 0</td>
</tr>
<tr>
<td>Binary*</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Bit</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
<tr>
<td>Blob*</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Boolean</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
<tr>
<td>Char*</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Clob*</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Date</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(precision to the nanosecond)</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
<tr>
<td>Long VarBinary*</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Long Varchar*</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Numeric</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>JDBC Datatype</td>
<td>Transformation</td>
<td>Range</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Real</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
</tbody>
</table>
| Smallint      | Integer        | -2,147,483,648 to 2,147,483,647   
|               |                | Precision 10, scale 0 |
| Time          | Date/Time      | Jan 1, 0001 A.D. to Dec 31, 9999 A.D.   
|               |                | (precision to the nanosecond) |
| Timestamp     | Date/Time      | Jan 1, 0001 A.D. to Dec 31, 9999 A.D.   
|               |                | (precision to the nanosecond) |
| Tinyint       | Integer        | -2,147,483,648 to 2,147,483,647   
|               |                | Precision 10, scale 0 |
| Varchar*      | String         | 1 to 104,857,600 characters |
| Varbinary*    | Binary         | 1 to 104,857,600 bytes |

*If the size of data in a port is greater than 100 MB, the Developer tool sets the port precision to 4000 by default. To process data with a larger size, increase the port precision.

---

**Microsoft SQL Server and Transformation Datatypes**

Microsoft SQL Server datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The following table compares Microsoft SQL Server datatypes and transformation datatypes:

<table>
<thead>
<tr>
<th>Microsoft SQL Server</th>
<th>Range</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>1 to 8,000 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Bit</td>
<td>1 bit</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Char</td>
<td>1 to 8,000 characters</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
</tbody>
</table>
| Datetime             | Jan 1, 1753 A.D. to Dec 31, 9999 A.D.   
|                      | Precision 23, scale 3   
|                      | (precision to 3.33 milliseconds) |
|                      | Date/Time                  | Jan 1, 0001 A.D. to Dec 31, 9999 A.D.   
|                      |                            | (precision to the nanosecond) |
| Decimal              | Precision 1 to 38, scale 0 to 38 | Decimal       | Precision 1 to 28, scale 0 to 28 |
| Float                | -1.79E+308 to 1.79E+308    | Double         | Precision 15               |
| Image                | 1 to 2,147,483,647 bytes  | Binary         | 1 to 104,857,600 bytes    |
### Microsoft SQL Server Datatypes

<table>
<thead>
<tr>
<th>Microsoft SQL Server</th>
<th>Range</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>-2,147,483,648 to 2,147,483,647</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Money</td>
<td>-922,337,203,685,477.5807 to 922,337,203,685,477.5807</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Numeric</td>
<td>Precision 1 to 38, scale 0 to 38</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Real</td>
<td>-3.40E+38 to 3.40E+38</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Smalldatetime</td>
<td>Jan 1, 1900, to June 6, 2079</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>Smallint</td>
<td>-32,768 to 32,768</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Smallmoney</td>
<td>-214,748.3648 to 214,748.3647</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Sysname</td>
<td>1 to 128 characters</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Text</td>
<td>1 to 2,147,483,647 characters</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Timestamp</td>
<td>8 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Tinyint</td>
<td>0 to 255</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>Varbinary</td>
<td>1 to 8,000 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Varchar</td>
<td>1 to 8,000 characters</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
</tbody>
</table>

### Unsupported Microsoft SQL Server Datatypes

The Developer tool does not support certain Microsoft SQL Server datatypes.

The Developer tool does not support the following Microsoft SQL Server datatypes:

- Bigint
- Nchar
- Ntext
- Numeric Identity
- Nvarchar
- Sql_variant

---

202 Appendix A: Datatype Reference
ODBC and Transformation Datatypes

ODBC datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The following table compares ODBC datatypes, such as Microsoft Access or Excel, to transformation datatypes:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Precision 19, scale 0</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Bit</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Char</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Date</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(precision to the nanosecond)</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
<tr>
<td>Long Varbinary</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Nchar</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Nvarchar</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Ntext</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Numeric</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>Real</td>
<td>Double</td>
<td>Precision 15</td>
</tr>
<tr>
<td>Smallint</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
<tr>
<td>Text</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Time</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(precision to the nanosecond)</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(precision to the nanosecond)</td>
</tr>
<tr>
<td>Tinyint</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precision 10, scale 0</td>
</tr>
</tbody>
</table>
Oracle and Transformation Datatypes

Oracle datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The following table compares Oracle datatypes and transformation datatypes:

<table>
<thead>
<tr>
<th>Oracle</th>
<th>Range</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blob</td>
<td>Up to 4 GB</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Char(L)</td>
<td>1 to 2,000 bytes</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Clob</td>
<td>Up to 4 GB</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Date</td>
<td>Jan. 1, 4712 B.C. to Dec. 31, 4712 A.D. Precision 19, scale 0</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>Long</td>
<td>Up to 2 GB</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you include Long data in a mapping, the Integration Service converts it to the transformation String datatype, and truncates it to 104,857,600 characters.</td>
</tr>
<tr>
<td>Long Raw</td>
<td>Up to 2 GB</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Nchar</td>
<td>1 to 2,000 bytes</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Nclob</td>
<td>Up to 4 GB</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Number</td>
<td>Precision of 1 to 38</td>
<td>Double</td>
<td>Precision of 15</td>
</tr>
<tr>
<td>Number(P,S)</td>
<td>Precision of 1 to 38, scale of 0 to 38</td>
<td>Decimal</td>
<td>Precision of 1 to 28, scale of 0 to 28</td>
</tr>
<tr>
<td>Oracle</td>
<td>Range</td>
<td>Transformation</td>
<td>Range</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
<td>----------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Nvarchar2</td>
<td>1 to 4,000 bytes</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Raw</td>
<td>1 to 2,000 bytes</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Jan. 1, 4712 B.C. to Dec. 31, 9999 A.D.</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D.</td>
</tr>
<tr>
<td></td>
<td>Precision 19 to 29, scale 0 to 9</td>
<td></td>
<td>(precision to the nanosecond)</td>
</tr>
<tr>
<td>Vvarchar</td>
<td>1 to 4,000 bytes</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>Vvarchar2</td>
<td>1 to 4,000 bytes</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>XMLType</td>
<td>Up to 4 GB</td>
<td>Text</td>
<td>1 to 104,857,600 characters</td>
</tr>
</tbody>
</table>

**Number(P,S) Datatype**

The Developer tool supports Oracle Number(P,S) values with negative scale. However, it does not support Number(P,S) values with scale greater than precision 28 or a negative precision.

If you import a table with an Oracle Number with a negative scale, the Developer tool displays it as a Decimal datatype. However, the Data Integration Service converts it to a double.

**Char, Vvarchar, Clob Datatypes**

When the Data Integration Service uses the Unicode data movement mode, it reads the precision of Char, Vvarchar, and Clob columns based on the length semantics that you set for columns in the Oracle database.

If you use the byte semantics to determine column length, the Data Integration Service reads the precision as the number of bytes. If you use the char semantics, the Data Integration Service reads the precision as the number of characters.

**Unsupported Oracle Datatypes**

The Developer tool does not support certain Oracle datatypes.

The Developer tool does not support the following Oracle datatypes:
- Bfile
- Interval Day to Second
- Interval Year to Month
- Mslabel
- Raw Mslabel
- Rowid
- Timestamp with Local Time Zone
- Timestamp with Time Zone
XML and Transformation Datatypes

XML datatypes map to transformation datatypes that the Data Integration Service uses to move data across platforms.

The Data Integration Service supports all XML datatypes specified in the W3C May 2, 2001 Recommendation. However, the Data Integration Service may not support the entire XML value range. For more information about XML datatypes, see the W3C specifications for XML datatypes at [http://www.w3.org/TR/xmlschema-2](http://www.w3.org/TR/xmlschema-2).

The following table compares XML datatypes to transformation datatypes:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>anyURI</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>base64Binary</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>boolean</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>byte</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>date</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>dateTime</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>decimal</td>
<td>Decimal</td>
<td>Precision 1 to 28, scale 0 to 28</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
<td>Precision of 15 digits</td>
</tr>
<tr>
<td>duration</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>ENTITIES</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>ENTITY</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>float</td>
<td>Double</td>
<td>Precision of 15 digits</td>
</tr>
<tr>
<td>gDay</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>gMonth</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>gMonthDay</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>gYear</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>gYearMonth</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>hexBinary</td>
<td>Binary</td>
<td>1 to 104,857,600 bytes</td>
</tr>
<tr>
<td>ID</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>IDREF</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>IDREFS</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
</tbody>
</table>
### Datatype

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Transformation</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>language</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>long</td>
<td>Bigint</td>
<td>-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>NCName</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>negativeInteger</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>NMTOKEN</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>NMTOKENS</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>nonNegativeInteger</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>nonPositiveInteger</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>normalizedString</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>NOTATION</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>positiveInteger</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>QName</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>short</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>string</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>time</td>
<td>Date/Time</td>
<td>Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)</td>
</tr>
<tr>
<td>token</td>
<td>String</td>
<td>1 to 104,857,600 characters</td>
</tr>
<tr>
<td>unsignedByte</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>unsignedShort</td>
<td>Integer</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
</tbody>
</table>

## Converting Data

You can convert data from one datatype to another.

To convert data from one datatype to another, use one of the following methods:

- Pass data between ports with different datatypes (port-to-port conversion).
Use transformation functions to convert data.
Use transformation arithmetic operators to convert data.

Port-to-Port Data Conversion

The Data Integration Service converts data based on the datatype of the port. Each time data passes through a port, the Data Integration Service looks at the datatype assigned to the port and converts the data if necessary.

When you pass data between ports of the same numeric datatype and the data is transferred between transformations, the Data Integration Service does not convert the data to the scale and precision of the port that the data is passed to. For example, you transfer data between two transformations in a mapping. If you pass data from a decimal port with a precision of 5 to a decimal port with a precision of 4, the Data Integration Service stores the value internally and does not truncate the data.

You can convert data by passing data between ports with different datatypes. For example, you can convert a string to a number by passing it to an Integer port.

The Data Integration Service performs port-to-port conversions between transformations and between the last transformation in a dataflow and a target.

The following table describes the port-to-port conversions that the Data Integration Service performs:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Bigint</th>
<th>Integer</th>
<th>Decimal</th>
<th>Double</th>
<th>String, Text</th>
<th>Date/Time</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigint</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Integer</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Decimal</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Double</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>String, Text</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Date/Time</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Binary</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
INDEX

A
abstract property
   schema object 81
active transformations
description 114
activities
   overview 149
application element
   parameter files 170
applications
   creating 160
   mapping deployment properties 159
   redeploying 163
   replacing 162
   updating 162
attribute properties
   schema object 84
attributeFormDefault
   schema object 85
attributes
   relationships 109

B
base property
   schema object 82
bigint
   constants in expressions 195
   high precision handling 194
   using in calculations 194
   writing to flat files 196
binary datatypes
   overview 196
block property
   schema object 82

C
certificates
   adding untrusted certificates 75, 89
   certificate properties 75, 89
   managing certificates 74, 88
   untrusted certificates 74, 88
cheat sheets
   description 9
collapse whitespace property
   schema object 82
column profile
   options 92
   overview 90
   process 91
column profile results
   column profile 95
complex type
   advanced properties 83
conditional sequence flows
   failed tasks 151
   overview 150
   task output 151
   workflow parameters 151
   workflow variables 151
conditions
   sequence flows 150
configurations
   troubleshooting 144
connections
   Connection Explorer view 29
   creating 36
   deleting 27
   editing 27
   IBM DB2 properties 29
   JDBC properties 30
   Microsoft SQL Server properties 31
   ODBC properties 32
   Oracle properties 33
   overview 27
   renaming 27
   web services properties 34
copy
   description 24
   objects 24
cost-based optimization
   description 182
custom queries
   Informatica join syntax 55
   left outer join syntax 57
   normal join syntax 56
   outer join support 55
   right outer join syntax 58
custom SQL queries
   creating 49
   customized data objects 48
customized data objects
   adding pre- and post-mapping SQL commands 59
   adding relational data objects 47
   adding relational resources 47
   advanced query 49
   creating 48
   creating a custom query 49
   creating key relationships 48
   creating keys 47
   custom SQL queries 48
   default query 49
   description 43
   entering source filters 52
   entering user-defined joins 54
   key relationships 45
   pre- and post-mapping SQL commands 58
   reserved words file 49
select distinct 52
simple query 49
sorted ports 53
troubleshooting 76
user-defined joins 54
using select distinct 52
using sorted ports 53
write properties 45

D
Data Integration Service
selecting 13
data object profiles
creating a single profile 94
data viewer
configuration properties 137
configurations 136, 139
creating configurations 140
troubleshooting configurations 144
database hints
entering in Developer tool 51
datatypes
BigInt 194
binary 196
Date/Time 196
decimal 197
double 197
flat file 198
IBM DB2 199
Integer 194
JDBC 200
Microsoft SQL Server 201
ODBC 203
Oracle 204
overview 193
port-to-port data conversion 208
string 198
transformation 194
XML 206
Date/Time datatypes
overview 196
decimal
high precision handling 194, 197
decimal datatypes
overview 197
default SQL query
viewing 49
dependencies
implicit 128
link path 128
deployment
mapping properties 159
overview 158
redeploying an application 163
replacing applications 162
to a Data Integration Service 161
to file 161
updating applications 162
Developer tool
workspace directory 6
double
high precision handling 197
double datatypes
overview 197

E
early projection optimization
description 181
early selection optimization
description 181
elementFormDefault
schema object 85
enumeration property
schema object 81
error messages
grouping 23
limiting 23
events
adding to workflows 149
overview 148
export
dependent objects 175
objects 176
overview 174
XML file 175
Expression Editor
description 118
validating expressions 119
expressions
adding comments 119
adding to a port 119
conditional sequence flows 150
entering 118
in transformations 117
pushdown optimization 187
validating 119

F
failed tasks
conditional sequence flows 151
find in editor
description 22
fixed value property
schema object 81
flat file data objects
advanced properties 68
column properties 60
configuring read properties 64
configuring write properties 67
creating 68
delimited, importing 70
description 59
fixed-width, importing 69
general properties 60
read properties 61, 65
folders
creating 20
description 20
full optimization level
description 180
functions
available in sources 187
pushdown optimization 187

gateways
adding to workflows 149
overview 149
generated prefix
changing for namespace 79

H
high precision
BigInt datatype 194
Decimal datatype 194
hints
Query view 51

I
IBM DB2 sources
pushdown optimization 186
identifying relationships
description 109
import
application archives 162
dependent objects 175
objects 178
overview 174
XML file 175
Informatica Developer
overview 5
profile views 91
setting up 9
starting 6
Informatica Marketplace
description 9
inherit by property
schema object 83
inherit from property
schema object 83
integers
constants in expressions 195
converting from strings 196
using in calculations 194
writing to flat files 196
Is Successful
output 151

J
join syntax
customized data objects 55
Informatica syntax 55
left outer join syntax 57
normal join syntax 56
right outer join syntax 56

K
key relationships
creating between relational data objects 42
creating in customized data objects 48
customized data objects 45
relational data objects 41

L
local workspace directory
configuring 6

logical data object mappings
creating 112
read mappings 112
types 112
write mappings 112
logical data object models
creating 101
description 101
element 100
importing 102
logical data objects
attribute relationships 109
creating 110
description 108
element 100
properties 109
logical view of data
developing 100
overview 99
logs
description 145
workflow instances 155

M
mapping object
running a profile 97
mapping parameters
overview 164
system 164
types 165
user-defined 164, 165
where to apply 167
where to create 166
mappings
adding objects 123
configuration properties 137
configurations 136, 141
connection validation 130
creating 123
creating configurations 142
deployment properties 159
developing 122
expression validation 130
object dependency 122
object validation 130
objects 123
optimization methods 181
overview 121
predicate optimization method 182
running 131
troubleshooting configurations 144
validating 130
validation 130
Mapplet and Mapping Profiles
Overview 97
mapplets
creating 135
input 134
output 135
overview 132
rules 134
types 133
validating 135
maximum length
schema object 81
maximum occurs
  schema object 81
member types
  schema object 82
Microsoft SQL Server sources
  pushdown optimization 186
pushdown optimization 186
minimal optimization level
  description 180
minimum length
  schema object 81
minimum occurs
  schema object 81
Model repository
  connecting 16
  description 14
  objects 14
monitoring
  description 146

N
namespace
  changing generated prefix 79
namespaces
  schema object 80
NaN
described 195
nillible property
  schema object 81
non-identifying relationships
  description 109
normal optimization level
  description 180

O
objects
  adding to workflows 149
  copying 24
operators
  available in sources 191
  pushdown optimization 191
  optimization
    cost-based optimization method 182
    early projection optimization method 181
    early selection optimization method 181
    mapping performance methods 181
    semi-join optimization method 183
  optimization levels
    description 180
Oracle sources
  pushdown optimization 186
outer join support
  customized data objects 55
overview
  transformations 114

P
parameter files
  application element 170
  creating 172
  mapping 164
  project element 169
  purpose 168
  running mappings with 168
  sample 170
  structure 168
  XML schema definition 168
parameters
  mapping 164
  passive transformations
    description 115
pattern property
  schema object 81
performance tuning
  cost-based optimization method 182
  creating data viewer configurations 140
  creating mapping configurations 142
  data viewer configurations 139
  early projection optimization method 181
  early selection optimization method 181
  mapping configurations 141
  optimization levels 180
  optimization methods 181
  predicate optimization method 182
  pushdown optimization 185
  semi-join optimization method 183
physical data objects
  customized data objects 43
  description 39
  flat file data objects 59
  relational data objects 40
  synchronization 75
  troubleshooting 76
port attributes
  propagating 127
ports
  connection validation 130
  linking 124
  linking automatically 125
  linking by name 126
  linking by position 126
  linking manually 125
  linking rules and guidelines 127
  propagated attributes by transformation 129
pre- and post-mapping SQL commands
  adding to relational data objects 59
  customized data objects 58
primary keys
  creating in customized data objects 47
  creating in relational data objects 41
profile results
  exporting in Informatica Developer 96
project element
  parameter files 169
project permissions
  allowing parent object access 19
  assigning 19
  dependent object instances 18
  external object permissions 18
  grant permission 17
  read permission 17
  showing security details 19
  write permission 17
projects
  assigning permissions 19
  creating 16
  description 16
  filtering 17
  permissions 17
  sharing 16
monitoring 155
overview 149
transformation datatypes
  list of 194
transformations
  active 114
  connected 115
  creating 119
  developing 116
  editing reusable 117
  expression validation 119
  expressions 117
  passive 115
  reusable 116
  unconnected 115

U
user-defined joins
  customized data objects 54
  entering 54
  Informatica syntax 55
  left outer join syntax 57
  normal join syntax 56
  outer join support 55
  right outer join syntax 58
user-defined parameters
  mapping 164

V
validation
  configuring preferences 23
  grouping error messages 23
  limiting error messages 23
  schema object 82
views
  Connection Explorer view 29

W
web service
  configuration properties 137
Welcome page
  description 8
workflow instances
  definition 155
  logs 155
  monitoring 155
  running 155
workflow parameters
  conditional sequence flows 151
workflow variables
  conditional sequence flows 151
workflows
  activities 149
  adding objects 149
  advanced properties 152
  conditional sequence flows 150
  creating 148
  deleting 155
  deploying 154
  events 148
  gateways 149
  instances 155
  logs 155
  monitoring 155
  overview 147
  running 155
  sequence flows 150
  tasks 149
  tracing level 152
  validation 153
workspace directory
  configuring 6
WSDL data objects
  advanced view 73
  importing 71
  overview view 72
  schema view 71
  synchronization 74