Developer Mapping Guide
This software and documentation contain proprietary information of Informatica LLC and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC. This Software may be protected by U.S. and/or international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013(d)(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging and Informatica Master Data Management are trademarks or registered trademarks of Informatica LLC in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright Direct Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Open Group. All rights reserved. Copyright © Liberty Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearcase Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright © Edifice, Inc. All rights reserved. Copyright © Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1998. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Sun Microsystems, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright © IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzo Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jifonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telsys Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha. All rights reserved. Copyright © JQWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright © MaxMind, Inc. All Rights Reserved. Copyright © T mate Software s.r.o. All rights reserved. Copyright © MapIt Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CRNI. All rights reserved.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at http://www.apache.org/licenses/. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an 'AS IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (http://www.mozilla.org/), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at http://www.gnu.org/licenses/lgpl.html. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright ©1993-2006, all rights reserved. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at http://www.openssl.org and http://www.openssl.org/source/license.html.

This product includes Perl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://cpan.org/docs/5.10.0.html. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://www.dom4j.org/ license.html.

The product includes software copyright 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://dojotoolkit.org/license.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://www.icu-project.org/repos/icu/icu/trunk/license.html.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at http://www.gnu.org/software/ kawa/Software-License.html.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at http://www.opensource.org/licenses/mit-license.php.

This product includes software developed by Boost (http://www.boost.org/) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at http://www.boost.org/LICENSE_1_0.txt.

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at http://www.pcre.org/license.txt.

# Table of Contents

**Preface** ................................................................. 11

Informatica Resources. ......................................................... 11

- Informatica My Support Portal. ........................................ 11
- Informatica Documentation. ........................................... 11
- Informatica Product Availability Matrixes. .......................... 11
- Informatica Web Site. .................................................... 12
- Informatica How-To Library. ............................................ 12
- Informatica Knowledge Base. .......................................... 12
- Informatica Support YouTube Channel. .............................. 12
- Informatica Marketplace. .............................................. 12
- Informatica Velocity. .................................................... 12
- Informatica Global Customer Support. ............................... 12

**Chapter 1: Mappings** .................................................. 14

Mappings Overview. ......................................................... 14

Mapping Objects. ............................................................ 15

- Object Dependency in a Mapping. .................................... 15
- Adding Objects to a Mapping. ......................................... 16

Developing a Mapping. ..................................................... 16

Creating a Mapping. .......................................................... 16

Linking Ports. ................................................................. 17

- One to One Links. ....................................................... 17
- One to Many Links. ..................................................... 17
- Manually Linking Ports. ................................................ 18
- Automatically Linking Ports. ......................................... 18
- Rules and Guidelines for Linking Ports. ............................. 19

Propagating Port Attributes. ............................................... 19

- Dependency Types. ..................................................... 20
- Link Path Dependencies. .............................................. 20
- Implicit Dependencies. ................................................ 20
- Propagated Port Attributes by Transformation. .................. 21

Mapping Run-time Properties. .......................................... 23

Target Load Order Constraints. ........................................ 26

- Target Load Order Tab. ................................................ 26
- Constraints with Insert and Delete Rows. .......................... 27
- Target Load Order Rules and Guidelines. ........................... 27
- Creating Constraints. ................................................... 28
- Target Load Order Example. .......................................... 28

Mapping Validation. .......................................................... 30

- Connection Validation. ................................................ 30
# Table of Contents

Expression Validation .......................................................... 30
Object Validation ............................................................... 30
Validating a Mapping .......................................................... 31
Running a Mapping ............................................................ 31
Segments ........................................................................ 31
Copying a Segment ............................................................ 31

**Chapter 2: Mapplets** ......................................................... 32
Mapplets Overview ............................................................ 32
Mapplet Types ................................................................. 32
Mapplets and Rules ............................................................ 33
Mapplet Input and Output .................................................... 33
  Mapplet Input .............................................................. 33
  Mapplet Output ............................................................ 34
Creating a Mapplet ............................................................ 34
Validating a Mapplet .......................................................... 34

**Chapter 3: Mapping Parameters** .......................................... 35
Mapping Parameters Overview ............................................... 35
System Parameters ............................................................ 36
User-Defined Parameters .................................................... 37
  Date/Time Parameters ..................................................... 38
Where to Create User-Defined Parameters ................................ 38
Where to Assign Parameters ................................................ 39
  Parameters in Expressions ............................................... 42
  Parameters in SQL Statements .......................................... 43
  Parameters for Relational Table Resources .................................. 44
  Parameters for Fields and Property Values .................................. 45
  Parameters for Port Lists .................................................... 46
Parameters in Mappings ....................................................... 46
  Parameter Instance Value .................................................... 47
Parameters in Mapplets ....................................................... 47
  Parameter Instance Values in Mapplets .................................... 48
  Mapplet Parameters in Mappings .......................................... 48
  Parameters in Mapplets Example .......................................... 49
Parameters in Logical Data Objects ......................................... 49
Parameters in Virtual Table Mappings ..................................... 50
Parameter Sets ................................................................. 51
  Run Mappings with Parameter Sets from Infacmd ......................... 52
Parameter Files ............................................................... 52
  Parameter File Structure .................................................... 52
  Project Element ............................................................. 53
  Application Element ........................................................ 54
Viewing Detailed Statistics for a Mapping Job. ............................................. 172
Viewing Logs for a Mapping Job. ................................................................. 173
Reissuing a Deployed Mapping Job. ............................................................. 173
Canceling a Mapping Job. ............................................................................ 174
Reject Files. ................................................................................................. 174
  Location of Reject Files. ........................................................................... 174
  Content of Reject Files. ............................................................................ 174

Chapter 8: Export to PowerCenter. ......................................................... 177
Export to PowerCenter Overview. ............................................................... 177
PowerCenter Release Compatibility. ............................................................ 178
  Setting the Compatibility Level. ............................................................... 178
Mapplet Export. ........................................................................................... 178
Mappings with Parameters Export. .............................................................. 179
Export to PowerCenter Options. ................................................................. 179
Exporting an Object to PowerCenter. .......................................................... 180
Export Restrictions. .................................................................................... 181
Rules and Guidelines for Exporting to PowerCenter. .................................. 183
Troubleshooting Exporting to PowerCenter. ............................................... 184

Chapter 9: Import From PowerCenter. ................................................... 185
Import from PowerCenter Overview. ......................................................... 185
Override Properties. .................................................................................. 186
Conflict Resolution. .................................................................................... 186
Import Summary. ......................................................................................... 186
Datatype Conversion. .................................................................................. 187
Transformation Conversion. ....................................................................... 187
  Transformation Property Restrictions. ....................................................... 188
Mappings with Parameters Import. ............................................................... 194
  Mappings with System-Defined Parameters Import. ................................ 195
PowerCenter Repository Connection Parameters. ...................................... 195
Connection Assignments. ........................................................................... 196
Importing an Object from PowerCenter. ..................................................... 197
Import Restrictions. .................................................................................... 198
Import Performance. ................................................................................... 198

Chapter 10: Performance Tuning. ............................................................. 199
Performance Tuning Overview. ................................................................... 199
Optimization Methods. ............................................................................... 200
  Early Projection Optimization Method. .................................................... 200
  Early Selection Optimization Method. ...................................................... 201
  Branch Pruning Optimization Method. ..................................................... 201
  Predicate Optimization Method. ............................................................... 201
Partitioned Transformations .................................................. 240
Restrictions for Partitioned Transformations .............................. 241
Cache Partitioning for Transformations ..................................... 241
Disable Partitioning for a Transformation ................................. 243
Maintain Order in a Partitioned Mapping ..................................... 243
Maintain a Stable Sort ............................................................... 244
Override the Maximum Parallelism for a Mapping ..................... 244
Suggested Parallelism for a Transformation ............................... 245
Execution Instances for Address Validator and Match Transformations .............. 246
Overriding the Maximum Parallelism Value .............................. 247
Troubleshooting Partitioned Mappings ..................................... 248

Index ......................................................................................... 249
Preface

Understand mapping concepts. Learn how to develop, run, and administer mappings. Create flexibility through mapping parameters and dynamic mappings. Optimize mappings through tuning and partitioning.

Informatica Resources

Informatica My Support Portal

As an Informatica customer, the first step in reaching out to Informatica is through the Informatica My Support Portal at https://mysupport.informatica.com. The My Support Portal is the largest online data integration collaboration platform with over 100,000 Informatica customers and partners worldwide.

As a member, you can:

- Access all of your Informatica resources in one place.
- Review your support cases.
- Search the Knowledge Base, find product documentation, access how-to documents, and watch support videos.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Documentation

The Informatica Documentation team makes 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 https://mysupport.informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. You can access the PAMs on the Informatica My Support Portal at https://mysupport.informatica.com.
Informatica Web Site

You can access the Informatica corporate web site at https://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 How-To Library

As an Informatica customer, you can access the Informatica How-To Library at https://mysupport.informatica.com. The How-To Library is a collection of resources to help you learn more about Informatica products and features. It includes articles and interactive demonstrations that provide solutions to common problems, compare features and behaviors, and guide you through performing specific real-world tasks.

Informatica Knowledge Base

As an Informatica customer, you can access the Informatica Knowledge Base at https://mysupport.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Support YouTube Channel

You can access the Informatica Support YouTube channel at http://www.youtube.com/user/INFASupport. The Informatica Support YouTube channel includes videos about solutions that guide you through performing specific tasks. If you have questions, comments, or ideas about the Informatica Support YouTube channel, contact the Support YouTube team through email at supportvideos@informatica.com or send a tweet to @INFASupport.

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.

Informatica Velocity

You can access Informatica Velocity at https://mysupport.informatica.com. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support. Online Support requires a user name and password. You can request a user name and password at http://mysupport.informatica.com.
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
- Operation mapping with an operation as the mapping input, output, or both
- Virtual table mapping with a virtual table as the mapping output

When the schema or metadata of a flat file or relational source frequently changes, develop a dynamic mapping to manage the schema or metadata changes at run time. You can also develop a dynamic mapping to reuse the mapping logic and run the mapping against different physical sources and targets. A dynamic mapping is a mapping in which you can change the sources, targets, and transformation logic at run time based on parameters and rules that you define.

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.

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.

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.

The following objects in a mapping are stored as dependent repository objects:

- Virtual tables. Virtual tables are stored as part of an SQL data service.
- Non-reusable transformations that you build within the mapping. Non-reusable transformations are stored within the mapping only.
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, reference table, 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.

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, logical data objects, or virtual tables 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.

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.

**RELATED TOPICS:**

- “Links in Dynamic Mappings” on page 109

**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.

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>Address Validator</td>
<td>None.</td>
<td>None. This transform has predefined port names and datatypes.</td>
</tr>
<tr>
<td>Aggregator</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Association</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Case Converter</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Classifier</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Comparison</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Consolidator</td>
<td>None.</td>
<td>None. This transform has predefined port names and datatypes.</td>
</tr>
<tr>
<td>Data Masking</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Data Processor</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Decision</td>
<td>- Downstream ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Expression</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Filter</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>- Port name</td>
</tr>
<tr>
<td>Hierarchical to Relational</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Joiner</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td>- Implicit Dependencies</td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Transformation</td>
<td>Dependency</td>
<td>Propagated Attributes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Key Generator</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Labeler</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Lookup</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Associated ports (dynamic lookup)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implicit Dependencies</td>
<td></td>
</tr>
<tr>
<td>Match</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Merge</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Normalizer</td>
<td>- Ports in link path</td>
<td>- Port name</td>
</tr>
<tr>
<td>Parser</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Rank</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST Web Service</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Consumer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Router</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Condition</td>
<td></td>
</tr>
<tr>
<td>Sequence Generator</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Sorter</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>SQL</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Standardizer</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Union</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Transformation</td>
<td>Dependency</td>
<td>Propagated Attributes</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Update Strategy</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td></td>
<td>- Expression</td>
<td>- Port name</td>
</tr>
<tr>
<td></td>
<td>- Implicit dependencies</td>
<td>- Datatype, precision, scale</td>
</tr>
<tr>
<td>Weighted Average</td>
<td>- Ports in link path</td>
<td>- Port name, datatype, precision, scale, description</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mapping Run-time Properties**

The mapping run-time properties depend on the execution environment that you select for the mapping.

Configure the following mapping run-time properties:

**Validation Environments**

Indicates whether the Developer tool validates the mapping definition for the native execution environment, the Hadoop execution environment, or both. When you run a mapping in the native environment, the Data Integration Service processes the mapping.

Based on your license, you can run a mapping in the Hadoop environment. When you run a mapping in the Hadoop environment, the Data Integration Service pushes the mapping execution to the Hadoop cluster through a Hadoop connection. The Hadoop cluster processes the mapping.

When you choose the Hadoop execution environment, you can select the Hive on MapReduce engine or the Blaze engine to process the mapping. You can also select the Hive version. Select a version number from the list or assign a parameter to the Hive version. Before you can assign it to the Hive version property, you must define a parameter in the Parameters view. The parameter must be a string that contains a version from the Hive Version list.

The following figure shows the validation environment:

Choose both validation environments if you want to test the mapping in the native environment before you run the mapping in the Hadoop environment. Or, choose both validation environments if you want to define the execution environment value in a parameter when you run the mapping.

If you choose both environments, you must choose the execution environment for the mapping in the run-time properties.

Default is native.
**Execution Environment**

Select the execution environment to use when the mapping runs. When you run a mapping in the native environment, the Data Integration Service processes the mapping. If you installed PowerCenter Big Data Edition, you can run a mapping in the Hive environment. When you run a mapping in the Hive environment, the Data Integration Service pushes the transformation logic to the Hadoop cluster through a Hive connection. The Hadoop cluster processes the data.

You can use a mapping parameter to indicate the execution environment. Configure a string parameter. Set the default value to "Native" or "Hive." When you select the execution environment, click Assign Parameter, and select the parameter that you configured.

The following image shows where to select the mapping execution environment:

![Image showing execution environment selection](image)

When you choose the execution environment, the Developer tool saves one of the associated validation environments for the mapping run.

**Maximum Parallelism**

Valid for the native execution environment. Maximum number of parallel threads that process a single mapping pipeline stage. An administrator sets maximum parallelism for the Data Integration Service to a value greater than 1 to enable mapping partitioning. The administrator sets the maximum parallelism in the Administrator tool.

The default maximum parallelism value for a mapping is Auto. Each mapping uses the maximum parallelism value defined for the Data Integration Service. You can change the default maximum parallelism value to define a maximum value for a particular mapping. When maximum parallelism is set to different integer values for the Data Integration Service and the mapping, the Data Integration Service uses the minimum value.

Default is Auto. Maximum is 64.

For more information about partitioning, see "Partitioned Mappings Overview" on page 228.

**Target Commit Interval**

The number of rows you want to use as a basis for a commit. The Data Integration Service commits data based on the number of target rows that it processes and the constraints on the target table. The Data Integration Service tunes the commit intervals. The default commit interval is 10,000 rows.

The commit interval is an approximate interval for the Data Integration Service to issue the commit. The Data Integration Service might issue a commit before, on, or after, the commit interval. In general, the Data Integration Service checks the target commit interval after writing a complete writer buffer block.

**Stop on Errors**

Stops the mapping if a nonfatal error occurs in the reader, writer, or transformation threads. Default is disabled.

The following types of errors cause the mapping to stop when you enable Stop on Errors:
Reader errors

Errors encountered by the Data Integration Service while reading the source database or the source files. Reader errors can include alignment errors while running a session in Unicode mode.

Writer errors

Errors encountered by the Data Integration Service while writing to the target database or to the target files. Writer errors can include key constraint violations, loading nulls into a not null field, and database trigger responses.

Transformation errors

Errors encountered by the Data Integration Service while transforming data. Transformation errors can include conversion errors and any condition set up as an ERROR, such as null input.

Mapping Impersonation User Name

Valid for the native execution environment. Enables another user to impersonate the Data Integration Service user that connects to Hive, HBase, or HDFS sources and targets that use Kerberos authentication.

Enter a user name in the following format: <Hadoop service name>/<hostname>@<YOUR-REALM>

Where:

- Hadoop service name is the name of the Hadoop service that the Hive, HBase, or HDFS source or target resides.
- Hostname is the name or IP address of the Hadoop service.
- YOUR-REALM is the Kerberos realm.

The following special characters can only be used as delimiters: '/' and '@'

Suggested Parallelism

Valid for the native execution environment when the Maximum Parallelism property is assigned to a value greater than 1 or to a parameter. Suggested number of parallel threads that process the transformation pipeline stage. When you define a suggested parallelism value for a specific transformation, the Data Integration Service considers the value when it determines the optimal number of threads for that transformation pipeline stage. You might want to define a suggested parallelism value to optimize performance for a transformation that contains many ports or performs complicated calculations.

Default is Auto, which means that the transformation uses the maximum parallelism value defined for the mapping. Maximum is 64.

For more information about defining a suggested parallelism value, see "Suggested Parallelism for a Transformation" on page 245.

Hive Connection

Valid for the Hive execution environment. A Hive connection defines the connection information that the Data Integration Service requires to push the mapping execution to the Hadoop cluster. Select the Hive connection to run the mapping in the Hadoop cluster. You can assign a user-defined parameter for the Hive Connection. Define the parameter on the Parameters view of the mapping.
Target Load Order Constraints

A target load order constraint restricts how the Data Integration Service loads and commits rows to two target instances related to each other in the same mapping.

In the Developer tool you can configure constraints to restrict the order that the Data Integration Service loads rows to target tables.

You can configure a constraint to force the Data Integration Service to load the data of a primary target instance completely before loading data to a secondary target instance. The tables that you define as the primary target and secondary target depend on the transactions in the input rows.

Consider the following scenarios for target load order constraints:

**Insert Rows to a Master and a Detail Target**

You might configure a target load order constraint when you are inserting rows to targets that have a primary key-foreign key relationship. Configure the target with the primary key as the primary target instance. Configure the target with the foreign key as the secondary target instance. The Data Integration Service can stage the data for the secondary target until it completes loading the primary target.

**Delete Rows from a Master and a Detail Target**

When you need to delete rows from targets with a primary key-foreign key relationship, you configure a different constraint. Configure the target with the foreign key as the primary target instance to delete the rows from the detail target first. Configure the target with the primary key as the secondary target instance.

**Insert Rows and Update Rows to the Same Relational Table**

You can configure a target load order constraint for a mapping that loads insert rows and update rows to a relational table from two separate transformations. Configure the constraint to restrict the Data Integration Service from loading the update rows until after it loads the insert rows.

You can configure multiple constraints in a mapping. The Data Integration Service determines the most efficient execution plan to load the targets without violating the constraints.

Target Load Order Tab

Define target load order constraints on the **Load Order** tab in a mapping. Access the **Load Order** tab by clicking in the mapping canvas.

The following image shows the Load Order tab:
The image shows three constraints. Each constraint contains the primary target and the secondary target. The constraints specify that the Data Integration Service must load the Customer_Master target before the Customer_Orders target. The Sales_Rep_Master must load before the Customer_Orders target. The Sales_Office_Master must load before the Sales_Rep_Master.

You can enter the constraints in any order. You could enter the following pairs of constraints to indicate the same restrictions as in the previous image:

<table>
<thead>
<tr>
<th>Primary</th>
<th>Secondary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales_Office_Master</td>
<td>Sales_Rep_Master</td>
</tr>
<tr>
<td>Sales_Rep_Master</td>
<td>Customer_Master</td>
</tr>
<tr>
<td>Customer_Master</td>
<td>Customer_Orders</td>
</tr>
</tbody>
</table>

### Constraints with Insert and Delete Rows

Target load order constraints do not have special handling to process insert, update, and delete rows in the same file.

When you need to process insert, update, and delete rows, you can configure a Router transformation to return the insert and update rows to a different target instance than the delete rows. Configure target load order constraints to specify the order in which to load the targets.

For example, you might have an Order_Header and an Order_Detail target. The Order_Detail table has an OrderID foreign key to the Order_Header table. You need to process inserts, updates, and deletes in both tables.

You can separate the insert and update rows from the delete rows using a Router transformation. You configure the following output groups from the Router transformation:

1. Order_Header insert and update rows
2. Order_Header delete rows
3. Order_Detail insert and update rows
4. Order_Detail delete rows

You might create the following constraints for loading these rows to the targets:

```
Group #4 before group #2
Group #2 before group #1
Group #1 before group #3
```

These constraints force the Data Integration Service to process the deletes in the Order_Detail before the deletes in the Order_Header. The Data Integration Service processes all the deletes before the insert and update rows. It processes the Order_Header inserts and updates before the Order_Detail inserts and updates.

### Target Load Order Rules and Guidelines

Consider the following rules and guidelines when you define target load order constraints:

- In the Developer tool, you can configure some target columns as primary keys or foreign keys. Load order constraints ignores these keys. If the targets have primary key-foreign key constraints, you must define the load order constraints.
- The Developer tool does not validate the load order constraints as you define them. The Developer tool validates the load order constraints when it validates the mapping.
• The Data Integration Service can stage the data to a local disk for the second target instance in a target load order constraint. When the mapping has multiple secondary target instances, the Data Integration Service loads the staged data to the targets without violating the constraints.

• Informatica strongly advises against using the staging files or the data within the files. Informatica is not responsible for corrupted data that is caused by customer alteration of the staging files or the data in the tables. The structure of the staging files might change between Informatica versions.

• The Data Integration Service loads one target instance and then another target instance without determining whether the rows are inserts, deletes, or updates. For target tables with primary-key foreign-key constraints, an orphan row is a row in the foreign key target that does not have a matching row in the primary key target. The Data Integration Service does not check for orphan rows. The Data Integration Service loads all of the rows in the order you specify in the load order constraint.

Creating Constraints

Create target load order constraints on the Load Order tab of a mapping.

1. Create a mapping that contains more than one target load.
2. Click the mapping canvas.
   The mapping Properties tabs appear.
3. Click the Load Order tab.
4. To enter a constraint, click New.
   The Developer tool creates a row for the constraint.
5. Click the Primary field.
   A list of the target instances in the mapping appears.
6. Select the target instance that you want to load first.
7. In the Secondary field, select the target instance to load second.
8. Enter as many constraints as you need.

Target Load Order Example

An organization processes customer orders twice a day. It receives the customer information and order information in the same transaction file. The organization needs to ensure that the mapping that processes the order file loads the customer information before it loads the orders.

A developer creates a mapping that returns the customer information to a Customer_Target table. The mapping returns the orders to an Orders_Target table. The primary key of the Customer_Master is the CustomerID. Each order in the Orders table has a foreign key to the CustomerID in the Customer_Master. The developer creates a target load order constraint. The constraint restricts the Data Integration Service from loading the orders until it completes loading the customer information to the target.

The following image shows the mapping:
A Normalizer transformation creates separate output groups for the customer and orders data. The Developer needs to ensure that the customer data loads to the target before the orders data.

The Data Integration Service can use different strategies to implement the target load order constraint. In this example, the Data Integration Service creates two pipelines to load the customer data and the order data to the target tables.

The following image shows the pipelines that load the customer data and the order data to the target tables:
In the first pipeline, the Data Integration Service loads the customers to the Customer_Master and it stages the orders to a local disk file. In the second pipeline, the Data Integration Service loads the staged orders to the Orders 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, rule, or virtual stored procedure. 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, rule, or virtual stored procedure 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, rule, or virtual stored procedure.

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, rule, or virtual stored procedure.
5. Click Edit > Paste.
This chapter includes the following topics:

- Mapplets Overview, 32
- Mapplet Types, 32
- Mapplets and Rules, 33
- Mapplet Input and Output, 33
- Creating a Mapplet, 34
- Validating a Mapplet, 34

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. If you add a Sequence Generator transformation to a mapplet, it must be 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.

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.
- Midstream. The mapplet contains an Input transformation and an Output transformation. It does not contain a data source for input or 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, Lookup transformations, and passive data quality transformations. It cannot contain any other type of transformation. For example, a rule cannot contain a Match transformation, as it is an active transformation.
• It does not specify cardinality between input groups.

Note: Rule functionality is not limited to profiling. You can add any mapplet that you validate as a rule to a profile in the Analyst tool. For example, you can evaluate postal address data quality by selecting a rule configured to validate postal addresses and adding it to a profile.

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.
Mapping Parameters Overview

A mapping parameter represents a constant value that you can change between mapping runs. Create parameters to rerun a mapping with different values. Use parameters to change the values of connections, file directories, expression components, port lists, port links, and task properties.

You can configure system parameters or user-defined parameters.

System parameters.

Built-in parameters for a Data Integration Service. System parameters define the directories where the Data Integration Service stores log files, cache files, reject files, source files, target files, and temporary files. An administrator defines the system parameter default values for a Data Integration Service in the Administrator tool.

User-defined parameters.

Parameters that you define in transformations, logical data objects, mappings, and workflows. Create user-defined parameters to rerun a mapping with different connection, flat file, cache file, temporary file, expression, ports, or reference table values.
You can use parameters to determine which generated ports to use in a dynamic mapping at run time. You can configure parameters to indicate which ports to link at run time. You can assign a parameter to change the data object in a Read, a Write, or a Lookup transformation.

You can override parameter values by assigning a parameter set or a parameter file to a mapping. A parameter set is a repository object that contains mapping parameter values. A parameter file is an XML file that contains parameter values. When you run a mapping with a parameter set or a parameter file, the Data Integration Service uses the parameter values defined in the parameter set or parameter file. These values override the default parameter values you configured in the transformation, the mapping, the mapplet, or the workflow.

For more information about workflow parameters, see the Informatica Developer Workflow Guide.

**Related Topics:**
- "Parameters in Dynamic Mappings" on page 120

### 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, log files, and temporary files.

Define the values of some of the system parameters in the execution options for the Data Integration Service. An Administrator can update the values from the Administrator tool. The Data Integration Service determines the values of other system parameters at run time. You cannot override system parameter values in a parameter file or a parameter set.

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. For example, when you create an Aggregator transformation, the cache directory system parameter is the default value assigned to the cache directory field in Informatica Administrator. If you want to use a different cache directory location, create a user-defined parameter and configure a default parameter value.

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>LogDir</td>
<td>String</td>
<td>Default directory for Mapping task log 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>
<tr>
<td>ApplicationName</td>
<td>String</td>
<td>Name of the application</td>
</tr>
<tr>
<td>ExecutionEnvironment</td>
<td>String</td>
<td>Hadoop or Native environment.</td>
</tr>
<tr>
<td>MappingName</td>
<td>String</td>
<td>Name of the mapping that is running.</td>
</tr>
<tr>
<td>System Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>MappingRunStartTime</td>
<td>Date/time</td>
<td>The start time of the mapping that is running.</td>
</tr>
<tr>
<td>ServiceName</td>
<td>String</td>
<td>The Data Integration Service name.</td>
</tr>
<tr>
<td>UserName</td>
<td>String</td>
<td>Name of the user that is running the mapping.</td>
</tr>
</tbody>
</table>

**User-Defined Parameters**

User-defined parameters represent constant values that you can change between mapping runs.

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 sets that set the connection name to the U.S. customers table, the Canadian customers table, and the Mexican customers table. Run the mapping with a different parameter set for each mapping run.

You can create the following types of parameters:

**Connection parameters**
- Informatica connection names.

**Date/time parameters**
- Dates.

**Expression**
- An expression that defines a join condition, a filter expression, or a lookup condition.

**Input Linkset**
- A set of ports to link in the Run-time Linking dialog box.

**Numeric parameters**
- Integer, bigint, decimal, and double parameters.

**Port**
- Name of a port to reference in an expression.

**Port List**
- A list of ports to include a group. You can use a port list parameter in the Aggregator transformation or the Rank transformation, for example.

**Resource**
- The table, view, or synonym name of a relational data object. When the resource name is parameterized, then the Data Integration Service uses the parameter value in the runtime query to fetch the object.

**Sort List**
- A list of ports to sort with a Sorter transformation. The list includes the port name and an indicator for ascending or descending sort sequence.
String

String parameters represent flat file names, directories, table names or run-time properties. Define string parameters with a precision of 32768 characters or less.

When you use a parameter to set a property value, you must use the correct parameter type for the property. For example, you cannot use a connection type parameter for a target file name. You must use a numeric parameter type if you are using the parameter in a numeric expression.

In relational data objects, you do not need to escape the dollar sign ($) in SQL overrides, filter conditions, join conditions. The Data Integration Service treats a field that begins with a dollar sign in an SQL statement as a parameter.

Date/Time Parameters

You can create date parameters and use the parameters in expressions.

You must define a date parameter in one of the following formats:

- MM/DD/RR
- MM/DD/YYYY
- MM/DD/YYYY HH24:MI
- MM/DD/RR HH24:MI
- MM/DD/RR HH24:MI:SS
- MM/DD/YYYY HH24:MI:SS
- MM/DD/RR HH24:MI:SS.NS
- MM/DD/YYYY HH24:MI:SS.NS

Where to Create User-Defined Parameters

You can create user-defined parameters in flat file data objects, transformations, custom data objects, mapplets, mappings, and workflows. After you create the parameters, you can assign the parameters to fields such as conditions, expressions, connections, directories, and file names.

When you create a parameter for a transformation, a logical data object, a mapplet, a mapping, or a workflow, the parameter applies to that object. For example, you create a parameter in a transformation. Then you add the transformation to a mapplet. You can either use the default parameter value from the transformation or you can create a mapplet parameter to override the transformation parameter value.

To override the transformation parameter default value, bind the mapplet parameter to the transformation parameter. Change the default value for the mapplet parameter. You can override the mapplet parameter with a mapping parameter.

The following list shows where you can create parameters:

- Workflow parameters
- Mapping parameters
- Mapplet parameters
- Logical data objects
- Transformation/data object parameters

You can set workflow parameter values and mapping parameter values at run time by configuring the parameter values in a parameter set or a parameter file.
You can create parameters at the same time that you assign parameters to fields and properties. When you assign a parameter to a field, you can create the parameter to use. You can also browse for a parameter that you created previously.

Maintain user-defined parameters in the Parameters tab of a transformation or of a data object. A mapping, mapplet, workflow, or logical data object also has a Parameters tab. You can add, change, and delete parameters on the Parameters tab.

Note: When you create parameters on the Parameters tab, do not include a leading dollar sign ($) in the parameter name.

Where to Assign Parameters

You can assign user-defined parameters and system parameters to fields. You must create the user-defined parameters before you can assign them to fields.

You can parameterize some properties in objects and transformations. If you can assign a parameter to a property, the option appears when you configure the property value.

You can parameterize properties of a reusable or nonreusable transformations. When you add a reusable transformation to a mapping, you can override the default value with a mapping parameter. If you parameterize a nonreusable transformation, the parameters are mapping parameters.

The Read and Write transformations are nonreusable transformations that you create from physical data objects. You can parameterize the physical data objects that you create the transformations from. You can also assign parameters to some properties in Read and Write transformations.

The following table lists the objects and the fields where you can assign parameters:

<table>
<thead>
<tr>
<th>Object</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>All transformations</td>
<td>Link resolution order</td>
</tr>
<tr>
<td>Association transformation</td>
<td>Cache file directory</td>
</tr>
<tr>
<td></td>
<td>Cache file size</td>
</tr>
<tr>
<td>Address Validator transformation</td>
<td>Casing style</td>
</tr>
<tr>
<td></td>
<td>Default country</td>
</tr>
<tr>
<td></td>
<td>Geocode data type</td>
</tr>
<tr>
<td></td>
<td>Global maximum field length</td>
</tr>
<tr>
<td></td>
<td>Line separator</td>
</tr>
<tr>
<td></td>
<td>Maximum result count</td>
</tr>
<tr>
<td></td>
<td>Optimization level</td>
</tr>
<tr>
<td></td>
<td>Standardize invalid addresses</td>
</tr>
<tr>
<td>Aggregator transformation</td>
<td>Cache directory</td>
</tr>
<tr>
<td></td>
<td>Expression elements. Not the full expression.</td>
</tr>
<tr>
<td></td>
<td>Group By</td>
</tr>
<tr>
<td>Bad Record Exception transformation</td>
<td>Lower Threshold</td>
</tr>
<tr>
<td></td>
<td>Upper Threshold</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Field</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Case Converter transformation</td>
<td>Reference table.</td>
</tr>
<tr>
<td>Consolidation transformation</td>
<td>Cache file directory&lt;br&gt;Cache file size</td>
</tr>
<tr>
<td>Customized data object</td>
<td>Connection&lt;br&gt;Data object&lt;br&gt;Owner&lt;br&gt;SQL Query elements&lt;br&gt;Table name</td>
</tr>
<tr>
<td>Decision transformation</td>
<td>Decision script.</td>
</tr>
<tr>
<td>Duplicate Record Exception transformation</td>
<td>Cache file directory&lt;br&gt;Lower Threshold&lt;br&gt;Upper Threshold</td>
</tr>
<tr>
<td>Expression transformation</td>
<td>Expression elements. Not the full expression.&lt;br&gt;Port selector</td>
</tr>
<tr>
<td>Filter transformation</td>
<td>Filter condition elements&lt;br&gt;Filter condition. Full expression.</td>
</tr>
<tr>
<td>Flat file data object</td>
<td>Control file directory&lt;br&gt;Control file name&lt;br&gt;Connection name&lt;br&gt;Default scale&lt;br&gt;Flat file delimiter&lt;br&gt;Merge file directory&lt;br&gt;Source file directory&lt;br&gt;Source file name&lt;br&gt;Output file name&lt;br&gt;Output file directory&lt;br&gt;Reject file directory&lt;br&gt;Target directory</td>
</tr>
<tr>
<td>Joiner transformation</td>
<td>Cache directory&lt;br&gt;Join condition elements&lt;br&gt;Port selector</td>
</tr>
<tr>
<td>Key Generator transformation</td>
<td>Cache file directory&lt;br&gt;Cache file size</td>
</tr>
<tr>
<td>Labeler transformation</td>
<td>Reference table</td>
</tr>
<tr>
<td>Lookup transformation not including the physical data objects for the lookup source</td>
<td>Data object. Nonreusable transformation.&lt;br&gt;Dynamic port rules. Nonreusable transformation.&lt;br&gt;Lookup condition. Full expression, nonreusable transformation.&lt;br&gt;Port selector. Nonreusable transformation.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Field</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Mapping</td>
<td>Hive version</td>
</tr>
<tr>
<td></td>
<td>Run-time environment</td>
</tr>
<tr>
<td></td>
<td>Maximum parallelism</td>
</tr>
<tr>
<td>Match transformation</td>
<td>Cache directory on the Match Output tab</td>
</tr>
<tr>
<td></td>
<td>Cache directory on the Match Type tab</td>
</tr>
<tr>
<td></td>
<td>Index directory on the Match Type tab</td>
</tr>
<tr>
<td></td>
<td>Persistence method</td>
</tr>
<tr>
<td></td>
<td>Threshold</td>
</tr>
<tr>
<td>Nonrelational data object</td>
<td>Connection</td>
</tr>
<tr>
<td>Rank transformation</td>
<td>Cache directory</td>
</tr>
<tr>
<td></td>
<td>Expression elements. Not the full expression.</td>
</tr>
<tr>
<td></td>
<td>Group by ports</td>
</tr>
<tr>
<td></td>
<td>Rank port</td>
</tr>
<tr>
<td>Read transformation</td>
<td>Connection</td>
</tr>
<tr>
<td></td>
<td>Data object</td>
</tr>
<tr>
<td></td>
<td>Owner name. Relational only.</td>
</tr>
<tr>
<td></td>
<td>Resource/table name. Relational only.</td>
</tr>
<tr>
<td>Relational Data Object</td>
<td>Filter condition elements</td>
</tr>
<tr>
<td></td>
<td>Join condition elements</td>
</tr>
<tr>
<td></td>
<td>PreSQL query elements</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL query elements</td>
</tr>
<tr>
<td></td>
<td>SQL override elements</td>
</tr>
<tr>
<td>Router transformation</td>
<td>Group filter condition elements.</td>
</tr>
<tr>
<td></td>
<td>Group filter condition. Full expression.</td>
</tr>
<tr>
<td>Sorter transformation</td>
<td>Sort key</td>
</tr>
<tr>
<td></td>
<td>Group by</td>
</tr>
<tr>
<td></td>
<td>Work directory</td>
</tr>
<tr>
<td>SQL transformation</td>
<td>Connection</td>
</tr>
<tr>
<td>Standardizer transformation</td>
<td>Reference table</td>
</tr>
<tr>
<td>Token Parser transformation</td>
<td>Reference table</td>
</tr>
<tr>
<td>Update Strategy transformation</td>
<td>Update strategy expression elements.</td>
</tr>
<tr>
<td></td>
<td>Update strategy expression. Full expression.</td>
</tr>
<tr>
<td>Write transformation</td>
<td>Data object</td>
</tr>
<tr>
<td></td>
<td>Link resolution order</td>
</tr>
<tr>
<td></td>
<td>Reject directory</td>
</tr>
<tr>
<td></td>
<td>Reject file name</td>
</tr>
</tbody>
</table>
**Parameters in Expressions**

You can configure parameters in expressions or conditions in transformations, such as the Aggregator transformation, Lookup transformation, Expression transformation, and the Filter transformation.

For example, configure a filter condition in the Filter transformation. Choose the ports and the parameters to include in the condition. Select the system parameters or the user-defined parameters to include in the filter condition.

The following image shows a Filter condition that includes the Employee port and the Employee_Filter parameter:

![Filter condition image]

You can use parameters in expressions in the same arguments that accept port names as arguments. You cannot use a parameter to replace a constant argument in an expression.

For example, consider the TO_DECIMAL expression that converts a string to a decimal value:

```
TO_DECIMAL( value [, scale] )
```

The scale argument must be a constant value in the expression.

The following valid expression contains a constant argument for scale:

```
TO_DECIMAL( Input_Port, 10 )
```

The following expression is not valid because it contains a user-defined parameter for the scale argument:

```
TO_DECIMAL( value, Employee_Filter )
```
A parameter cannot contain another parameter. For example, if you configure Parameter1 and Parameter2 in a transformation, you cannot set the default value of Parameter1 to $Parameter2. If you nest the parameters, the mapping fails with a validation error at runtime.

**Expression Parameters**

You can configure an expression parameter type. An expression parameter is a parameter that contains a complete expression. You can use an expression parameter in a Filter transformation and a Lookup transformation.

Define an expression parameter in the Expression Editor. Select **Specify by Parameter** to indicate that the complete expression is parameterized.

The following image shows the **Specify by Parameter** option for the filter condition:

When you use an expression parameter, you can create the expression parameter or you can select an existing expression parameter to use in the transformation. An expression parameter can contain ports, operators, and constants. It cannot contain other parameters.

For example, in a Filter transformation you might create a filter expression parameter with the following default value: EmployeeID > 100. In the mapping, you might create a different expression parameter with the following default value: Dept < 2000. If you bind the mapping parameter to the transformation parameter, you can override the mapping expression parameter at run time. You might create expression parameters with different port names and operators for dynamic mappings.

**Parameters in SQL Statements**

You can include parameters in SQL statements that you add to relational data objects or to Lookup transformations.

The following image shows how you can parameterize an SQL query that reads from a relational source:
The Data Integration Service escapes the data in each parameter with a single quote (') when it expands a query. For example, you might have a SQL statement with a date parameter called $date_parm:

\[
\text{select * from <table_name> where <date_port> >$date_parm}
\]

The query might expand to  
\[
\text{select * from <table_name> where <date_port> > '01/31/2000 00:00:00'}
\]

Do not use a single quote in the default value for a parameter. You might get unexpected results.

A parameter name cannot contain a period (.) . An SQL query is not valid if it has a parameter that contains a period. For example, the following SQL statement has a parameter name that contains a period:

\[
\text{SELECT $\text{tname.ID, "MY_SOURCE"."NAME" FROM "MY_SOURCE" where FIELDX=1}}
\]

When you validate the query, the Data Integration Service returns an error that it cannot find the tname.ID parameter.

Parameters for Relational Table Resources

You can parameterize the resource name, the table owner, and the connection for a Read transformation. The resource is the table, view, or synonym name of the relational data object.

You might parameterize the resource name if you need to process multiple tables from the same database in a dynamic mapping.

Select the Read transformation in the mapping. In the Run-time tab of the Properties view, click the Value column to assign a parameter for a connection, table owner, or resource.

The following image shows where to assign the parameters for the connection, the resource name, and the table owner for the Read transformation:
Parameters for Fields and Property Values

You can configure parameters for some field or property values in transformations and physical data objects.

You can configure connection names for relational data objects, customized data objects, and Lookup transformations. In a flat file data object, you can configure parameters for input and output file directories and the reject file directory. You can also configure a parameter to change the flat file delimiter type.

The following image shows the parameter for the flat file delimiter on the Advanced tab of the physical data object:
Parameters for Port Lists

You can create parameters that contain lists of ports. You can reference these parameters in transformations such as the Sorter transformation, Rank transformation, Joiner transformation, and Expression transformation.

You can configure the following types of parameters that contain multiple port names:

**Port list**
A list of port names separated by commas. A port list parameter has the following syntax:
Port1,Port2,Port3

**Sort list**
A list of port names and the sort type for each port. The sort list parameter has the following syntax:
Port1:A,Port2:A,Port3:D

**Input linkset**
A set of ports to link at run time. The link set parameter contains name-value pairs in the following syntax:
Port1:=Port2, Port3:=Port4

Parameters in Mappings

If you define a parameter in a resusable transformation or data object, you can override the parameter default value when you add the transformation to a mapping. Configure a mapping parameter to override the parameter value in a specific mapping.

When you define a mapping parameter, you can bind the mapping parameter to a specific transformation parameter. The mapping parameter value overrides the default parameter value in the transformation.

When you bind a mapping parameter to a transformation parameter, the parameters must be the same type. The mapping parameter name does not have to be the same as the transformation parameter name.

You can use a parameter set or a parameter file to set the mapping parameter values at run time. You cannot set transformation parameter values with a parameter set or parameter file. You must configure mapping parameters if you want to change parameter values at run time.

Use one of the following methods to define mapping parameters:

**Define mapping parameters on the Parameters tab of the mapping Properties view**
On the mapping Parameters tab, you can manually enter each parameter name, the parameter attributes, and the default value. You can bind these parameters to transformation parameters whenever you add a transformation to the mapping. You can update the mapping parameters on the mapping Parameters tab.

**Add mapping parameters from transformation parameters**
After you add a transformation to a mapping, you can create a mapping parameter directly from the transformation Parameters tab. You can expose the transformation parameter as a mapping parameter. The Developer tool creates a mapping parameter that has the same properties as the transformation parameter.

**Add parameters to a nonreusable transformation**
If you create a transformation in a mapping, the transformation is a nonreusable transformation. If you parameterize any of the transformation properties, you create mapping parameters instead of transformation parameters.
Parameter Instance Value

When you add a reusable transformation with parameters to a mapping, you can configure the instance value for each parameter in the transformation.

The instance value is the parameter value for a specific mapping. You can set the instance value to a default value, a specific value, or to a mapping parameter value.

A mapping parameter or a mapplet parameter can override the default value of the transformation parameter. Select a mapping parameter or a mapplet parameter and bind the parameter to the transformation parameter.

Set the instance value on the transformation Parameters tab in the Properties view.

Choose one of the following options for the Instance Value:

**Expose as mapping parameter**
Create a mapping parameter with the same attributes as the transformation parameter. Bind the mapping parameter to the transformation parameter in the same step.

**Parameter**
Browse for and select a mapping parameter to bind to the transformation parameter. You can also create a mapping parameter and bind it to the transformation parameter. When you create the mapping parameter and bind it, you are performing the same task as the Expose As Mapping Parameter option. However, when you manually create the mapping parameter, you can configure a different name than the transformation parameter.

**Use default**
Use the default value from the transformation parameter. Skip binding a mapping parameter to the transformation parameter.

**Value**
Enter a default parameter value to use in the mapping. Skip binding a mapping parameter to the transformation parameter.

Parameters in Mapplets

You can bind a mapplet parameter to a parameter in a data object or in a transformation that is in the mapplet.

When you define a mapplet parameter, you can bind the mapplet parameter to a specific transformation parameter. The mapplet parameter value overrides the default parameter value in the transformation. When you bind a mapplet parameter to a transformation parameter, the parameters must be the same type. The mapplet parameter name does not have to be the same as the transformation parameter name. You can bind a mapplet parameter to more than one transformation parameter.

Use one of the following methods to define mapplet parameters:

**Define mapplet parameters on the Parameters tab of the mapplet Properties view**
On the mapplet Parameters tab, you can manually enter each parameter name, parameter attributes, and default value.
Add mapplet parameters from transformation parameters

After you add a transformation to a mapplet, you can create the mapplet parameter directly from the transformation Parameters tab.

Parameter Instance Values in Mapplets

When you add a reusable transformation with transformation parameters to a mapplet, you can set the instance value for each parameter. The instance value of the parameter is the parameter value in a specific mapplet.

After you add the transformation to a mapplet, set the instance value on the transformation Parameters tab.

Choose one of the following options for the Instance Value:

Expose as mapplet parameter

Create a mapplet parameter with the same attributes as the transformation parameter. Bind the mapping parameter to the transformation parameter in the same step.

Parameter

Bind a mapplet parameter to the transformation parameter. You can browse for and select a mapplet parameter to bind to the transformation parameter. You can also create a mapplet parameter and bind that parameter to the transformation parameter. When you create a mapplet parameter and bind it, you are performing the same task as the Expose As Mapplet Parameter option. However, when you manually create the mapplet parameter, you can configure it with a different name and default value than the transformation parameter.

Use default

Use the default value from the transformation parameter. Skip binding a mapplet parameter to the transformation parameter.

Value

Enter a different default parameter value to use in the mapplet. Skip binding a mapplet parameter to the transformation parameter.

Mapplet Parameters in Mappings

When you add a mapplet with mapplet parameters to a mapping, you can set the instance values for the mapplet parameters. The instance value of a mapplet parameter is the parameter value for a specific mapping.

Set the instance value on the mapplet Parameters tab in the Properties view.

Choose one of the following options for the Instance Value:

Expose as mapping parameter

Create a mapping parameter with the same attributes as the mapplet parameter. Bind the mapping parameter to the mapplet parameter in the same step.

Parameter

Bind a mapping parameter to the mapplet parameter. You can browse for and select a mapping parameter to bind to the mapplet parameter. You can also create a mapping parameter and bind it to the mapplet parameter. When you create a mapping parameter and bind it, you are performing the same task as the Expose As Mapping Parameter option. However, when you manually create the mapping parameter, you can configure it with a different name and default value than the mapplet parameter.
Parameters in Mapplets Example

You can define mapplet parameters and override them with mapping parameters.

You might define an SQL transformation that returns customer data from a Customer table. You add the SQL transformation to a mapplet and parameterize the run-time connection.

You then add the mapplet to mappings that retrieve customer data from different databases. You define a mapping parameter in each mapping to override the default connection from the mapplet parameter.

The following table lists the connection parameters that you might create for the mapplet and for the mappings:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Object Type</th>
<th>Parameter Name</th>
<th>Parameter Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp_Get_Customer</td>
<td>Mapplet</td>
<td>mp_cust_connection</td>
<td>Oracle_Default</td>
</tr>
<tr>
<td>m_billing_mapping</td>
<td>Mapping</td>
<td>m_acctg_connection</td>
<td>Oracle_AcctDB</td>
</tr>
<tr>
<td>m_order_fill_mapping</td>
<td>Mapping</td>
<td>m_shipping_connection</td>
<td>Oracle_Warehouse</td>
</tr>
<tr>
<td>m_cust_main_mapping</td>
<td>Mapping</td>
<td>m_master_connection</td>
<td>Oracle_Cust_Mast</td>
</tr>
</tbody>
</table>

The mapplet, mp_Get_Customer, has a connection parameter called mp_cust_connection. The parameter has Oracle_Default as the default connection name. This connection might reference a test database, for example.

Each mapping has a connection parameter that overrides the mp_cust_connection parameter. Each mapping connects to the accounting, the warehouse, or the customer master database.

You must bind each mapping parameter to the mapplet parameter to override the default value. To change the value of the mapping parameters at runtime, you can configure a parameter set or a parameter file.

Parameters in Logical Data Objects

You can include parameters in logical data objects. You can use them in transformations and in the Read and Write mappings.

A logical data object can have a Read mapping and a Write mapping. A Read or Write mapping can contain transformations that use parameters. You can bind the reusable transformation parameters to parameters in the Read or Write mapping.

For example, a logical data object has a Read mapping that contains an Expression transformation. The Expression transformation has a parameter that defines a decimal value in an expression. The default value is 100.
When you add the Expression transformation to the Read mapping, you might want to use a different parameter value. You can create a parameter at the Read mapping level to override the transformation parameter. Click **Expose as Mapping Parameter** to create a duplicate parameter in the Read mapping. The Developer tool binds the duplicate parameter to the transformation parameter.

The following image shows the **Parameters** tab for the Expression transformation in the Read mapping:

![Parameters tab for the Expression transformation in the Read mapping](image)

To view the duplicate parameter, click the editor to display the mapping **Properties** view. Click the **Parameters** tab. You can change the parameter default value at the Read mapping level.

When you add the logical data object to a mapplet or mapping, you can override the Read mapping parameter. Create a duplicate parameter in the mapplet or the mapping. Change the default value of the duplicate parameter.

### Parameters in Virtual Table Mappings

A virtual table mapping defines the data flow between sources and a virtual table in an SQL data service. A virtual table mapping can contain parameters, but you cannot use a parameter file or a parameter set to override the parameter default values.

A virtual table mapping might contain reusable transformations or mapplets that contain parameters. You can bind mapping parameters to the transformation or mapplet parameters in a virtual table mapping. However, when a virtual table mapping contains parameters, the Data Integration Service applies the default parameter values from the mapping level. The Data Integration Service cannot bind values from a parameter file or from a parameter set to parameters in a virtual table mapping.

You can use a parameterized source that is connected to a virtual table mapping. The mapping uses the default parameter value.
Parameter Sets

A parameter set is an object in the Model repository that contains a set of parameters and parameter values to run mappings and workflows.

When you create a parameter set, you choose a mapping or workflow to use the parameters. After you choose a mapping or workflow, you can manually enter parameters in the parameter set or you can select parameters that are already in the repository for the mapping or the workflow.

You can use parameter sets for different situations. For example, you might use a specific parameter set when you run a workflow in a test environment.

You use a parameter set with a mapping, Mapping task, or workflow. You can add one or more parameter sets to an application when you deploy the application. You can add a parameter set to multiple applications and deploy them. To use a parameter set with a workflow or mapping, you must add the parameter set to the application when you deploy the workflow or mapping.

The following image shows a parameter set that contains parameters for two mappings:

The parameter set contains the following information:

**Object Name**

The name of the mapping, mapplet, or workflow that contains the parameter definition.

**Parameter Name**

The name of the parameter in the mapping, mapplet, or workflow.

**Value**

The value of the parameter to use at runtime. The value of the parameter in the parameter set overrides the parameter value in the mapping or workflow.

**Type**

The type of the parameter. Example parameter types include strings, numeric types, connections, port lists, sort lists, and date/time parameters.
Run Mappings with Parameter Sets from Infacmd

You can deploy a mapping as an application and include a parameter set in the application. You can then run the deployed application and use the parameter set.

After you deploy the mapping, you can run the deployed mapping with the parameter set from the command line. If you need to use different parameter sets, you can deploy more than one parameter set in the application. When you run the mapping you can specify which parameter set to use.

After the application is deployed, you can add parameter set entries with the infacmd addParameterSetEntries command. You can update parameter set entries with the infacmd updateParameterSetEntries command.

For more information about using parameter sets with infacmd, see the Informatica Command Reference.

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 from the command line.

The parameter values define properties for a mapping or a mapplet. The Data Integration Service applies these values when you run a mapping from the command line and specify a parameter file.

You can define mapping parameters and workflow parameters in 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 then 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 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 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, or workflow 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.
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 can include a workflow or mapping. You cannot include a transformation or data source in 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>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>
</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.

For example, you want the Data Integration Service to apply parameter values when you run mapping "MyMapping".

```
<project name="MyProject">
   <!-- Apply this parameter value to mapping "MyMapping" in project "MyProject". -->
   <mapping name="MyMapping">
      <parameter name="MyMapping_Param">Param_value</parameter>
   </mapping>
</project>
```
Application Element

An application element provides a run-time scope for a project element. An application element defines the parameter values to use when you run a specific mapping in a specific deployed application.

An application element defines the deployed application that contains objects that use parameters. An application element can contain a mapping element that defines a mapping in the deployed application that uses parameters. A mapping element contains a project element.

For example, you want the Data Integration Service to apply parameter values when you run mapping "MyMapping" in deployed application 'MyApp.' You do not want to use the parameter values when you run the mapping in any other application or when you run another mapping in project "MyProject." Define the parameters within the following elements:

```xml
<application name="MyApp">
  <project name="MyProject">
    <mapping name="MyMapping">
      <parameter name ="MyMapping_Param">Param_value</parameter>
    </mapping>
  </project>
</application>
```

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
  <parameter name="Param1" />
  ```
- Within an element, artifact names are not case-sensitive. Therefore, the Data Integration Service interprets <parameter name="SrcDir"/> and <parameter name="Srcdir"/> as the same application.
- A parameter that identifies a reference table must use a forward-slash (/) to separate folder names in a repository folder path.

Sample Parameter File

The following example shows a sample parameter file used to run mappings.

```xml
<?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 "Appl."

  This section assigns values to parameters created in mappings "Map1" and "Map2."
-->
  <application name="Appl1">
    <mapping name="Map1">
      <project name="Project1">
        <mapping name="Map1">
```
<parameter name="MAP1_PARAM1">MAP1_PARAM1_VAL</parameter>
<parameter name="MAP1_PARAM2">MAP1_PARAM2_VAL</parameter>
</mapping>
</project>
</mapping>
<project name="Map2">
<map name="Map2">
<parameter name="MAP2_PARAM1">MAP2_PARAM1_VAL</parameter>
<parameter name="MAP2_PARAM2">MAP2_PARAM2_VAL</parameter>
</map>
</project>
</application>
</!--
The Data Integration Service uses this section only when you run mapping "Map1" in project "Project1" in deployed application "App2."

This section assigns values to parameters created in the following objects:

* Mapping "Map1"

-->  
<application name="App2">
<map name="Map1">
<project name="Project1">
<map name="Map1">
<parameter name="MAP1_PARAM2">MAP1_PARAM2_VAL</parameter>
</map>
</project>
</map>
</application>

<!--
The Data Integration Service uses this section when you run any mapping that includes maplet "DS1" in project "Project1."

This section assigns values to parameters created in the following objects:

* Maplet "DS1"

-->  
<project name="Project1">
<maplet name="DS1">
<parameter name="PROJ1_DS1">PROJ1_DS1_VAL</parameter>
<parameter name="PROJ1_DS1_PARAM1">PROJ1_DS1_PARAM1_VAL</parameter>
</maplet>
</project>

-->  
<project name="Project2">
<transformation name="TX2">
<parameter name="RTL\PATH">Project1\Folder1\RTL1</parameter>
</transformation>
<folder name="Folder2">
<maplet name="MPLT1">
<parameter name="PROJ2_FOLD2_MPLT1">PROJ2_FOLD2_MPLT1_VAL</parameter>
</maplet>
</folder>
</project>

Parameter Files
Creating a Parameter File

The infacmd ms ListMappingParams command lists the parameters for a mapping in a deployed application and the default value for each parameter. Use the output of this command to create a parameter file.

1. Run the infacmd ms ListMappingParams command to list the parameters for 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":
   infacmd ms ListMappingParams -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a MyApplication -m MyMapping -o MyOutputFile.xml
   The Data Integration Service lists all mapping parameters with their default values.

2. If you did not specify the -o argument, you can 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.

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":
   The Data Integration Service fails the mapping when you run it with a parameter file and the parameter file is not valid. The Data Integration Service fails the mapping if it cannot find the parameter file or it cannot access the parameter file.

How to Configure Parameters

Define parameters in a transformation, a mapping, a mapplet, or a workflow.

The following image shows the process to use parameters in a reusable transformation in a mapping.
1. In a reusable transformation, create a parameter for a property in the transformation or for a variable in the Expression Editor.

2. Add the transformation to a mapping or to a mapplet.

3. In the transformation Parameters tab, choose how to set the parameter value in the mapping or the mapplet.
   - Expose the transformation parameter as a mapping parameter. Creates a duplicate of the transformation parameter at the mapping level.
   - Bind the transformation parameter to a mapping parameter. Browse for a mapping parameter or manually create a mapping parameter to bind to the transformation parameter.
   - Enter a specific parameter value. Enter a default value to use in the mapping run.
   - Use the default value of the transformation parameter. Use the original parameter value in the mapping.

After you bind a mapping parameter to the transformation parameter, you can create parameter sets to override the mapping parameter value at run time. Run the mapping from the command line and specify which parameter set to use for that mapping run.

Creating a Parameter for a Transformation Property

When you assign a parameter to field or to a transformation property, you can browse for a parameter to use or you can create a parameter specifically for the field.

1. Navigate to the field or property that you want to update.

2. Click the selection arrow in the Value column.

   If you can parameterize the property, the Assign Parameter option appears.

   The following image shows the Assign Parameter option for the Cache Directory:
3. Click **Assign Parameter**.

   The **Assign Parameter** dialog box appears. The dialog box shows the system parameters and the user-defined parameters that you created in the transformation.

   The following image shows the **Assign Parameter** dialog box:

![Assign Parameter dialog box](image1)

4. To create a parameter, click **New**.

5. Enter the parameter name, the type, the precision, and the default value.

   The following image shows the parameter called MyCacheParameter in the **Parameters** dialog box:

![Parameters dialog box](image2)
6. Click **OK**.

The parameter name appears in the transformation property.

The following image shows MyCacheParameter in the Aggregator transformation cache directory:

![Parameter Configuration](image)

You can add, change, and delete parameters in the transformation **Parameters** tab.

### Creating a Parameter in an Expression

You can reference the parameter in an Expression after you define the parameter. The following example shows how to assign a parameter to a component in a filter expression.

1. In the Filter transformation, click the **Filter** tab.
   
The Expression Editor appears. You can select functions, ports, and parameters to create the expression.

2. Select **Specify By Value** to define the expression instead of using an expression parameter.
3. On the Filter tab, click the **Ports** tab.

4. Select the Amount port. On the **Functions** tab, select the greater than (>) function.

   The following image shows the expression that contains the Amount port and the > operator:

5. Click the **Parameters** tab in the Expression Editor.

   The Expression Editor lists the system parameters and the user-defined parameters.

6. Click **New Parameter** to add a parameter.

   The **Parameters** dialog box appears with default values in the parameter attributes.

7. Enter the parameter name, parameter type, precision, and default value.

   The following image shows the **Parameters** dialog box:
8. In the Expression Editor, click **OK**

The parameter that you created appears in the parameter list.
9. Select the Min_Order_Amount parameter to add it to the expression. The Min_Order_Amount parameter appears in the expression.

The parameter appears in the expression with a dollar sign ($) identifier. The Min_Order_Amount default value is 50. If you add the transformation to a mapping without overriding the Min_Order_Parameter, the Filter transformation returns rows where the Amount is greater than 50.

**Expose Transformation Parameters as Mapping Parameters**

After you add a transformation to a mapping, you can expose a transformation parameter as a mapping parameter. When you expose a transformation parameter as a mapping parameter, you create a mapping parameter that is a duplicate of the transformation parameter.

The instance value of a parameter is the parameter value to use in a specific mapping or mapplet. When you expose a transformation parameter as a mapping parameter, you create a mapping parameter and bind it to a transformation parameter in one step. You can set mapping parameter values at run time with a parameter set or parameter file.

1. Open the mapping. Click the transformation that contains a parameter.
   The Properties view for the transformation appears.
2. Click the Parameters tab.
The following image shows the Parameters tab for a Filter transformation:

3. To create a mapping parameter for the parameter, select the parameter and click *Expose as Mapping Parameter*.
   The Developer tool creates a mapping parameter with the same name and it binds it to the transformation parameter.

4. To update a mapping parameter, open the Parameters tab of the mapping Properties view.
   You can change the default mapping parameter values. You can also add mapping parameters on the Parameters tab.

### Setting the Parameter Instance Value

You can set the parameter instance value from the *Instance Value* column on the transformation Parameters tab. Set the instance value in this column if you do not want to create a duplicate mapping parameter.

You can set a transformation parameter to a default value or you can bind an existing mapping parameter to the transformation parameter.

1. After you add a transformation to a mapping, click the Parameters tab on the Properties view of the transformation.
2. To bind a mapping parameter to a transformation parameter, perform the following steps:
   a. Click the *Instance Value* column for the transformation parameter.
      The Specify By dialog box appears.
   b. Click *Specify By Parameter*.
   c. On the Assign Parameter dialog box, browse for and select a mapping parameter or a system-defined parameter to bind to the transformation parameter.
   d. Click OK.
      The mapping parameter name appears as the parameter value in the Specify By dialog box.
   e. Click OK in the Specify By dialog box.
      The mapping parameter name appears in the *Instance Value* column.
3. To set a default value for the transformation parameter instance, use the following steps:
   a. Click the **Instance Value** column for the transformation parameter.
      The **Specify By** dialog box appears.
   b. To enter a default value, click **Specify By Value** and enter a default value for the instance.
   c. To use the transformation parameter default value, click **Use Default**.

### Creating a Parameter Set

Create a parameter set that you can use to change the runtime context for mappings and workflows.

When you create the parameter set, choose a mapping or workflow to contain the parameters. After you choose a mapping or workflow, you can manually enter parameters in the parameter set or you can select parameters.

1. In the Object Explorer view, right-click a project and click **New > Parameter Set**.
2. Enter a name for the parameter set and click **Finish**.

3. Drag the **Properties** panel down and view the grid to add the parameters to the parameter set.
4. Click **New > Mapping/Workflow**.
5. In the **Add Parameters** dialog box click **Browse** to find the mapping or workflow that contains the parameters you need to include in the set. A list of mappings and workflows appears.

6. Select a mapping or a workflow and click **OK**. A list of parameters from the mapping or workflow appears.

7. Select the parameters to include in the parameter set and then click **OK**. The mapping or the workflow name and the path appears in the parameter set. Each parameter that you selected appears beneath the object.
8. To add a parameter that is not yet in a workflow or mapping, right-click a mapping or object name and select Parameter insert.

The Developer tool creates a parameter beneath the workflow or mapping. Change the parameter name, the value, and the type.

**Note:** You must add the parameter to the mapping or workflow before you use the parameter set.
Mapping Outputs Overview

A mapping can return mapping outputs. A mapping output is a single value that is the result of aggregating a field or expression from each row that the mapping processes.

A mapping output returns a value that provides information about the mapping run. For example, a mapping output can return the number of error rows that the mapping found. A mapping output can return the latest order date that the mapping processed and the total amount of all the orders.

Transformations do not receive the mapping output values. The mapping returns each mapping value when the mapping completes. You can pass mapping outputs to other tasks in the workflow. You can save the values to use as input parameters the next time a mapping runs. You can define multiple mapping outputs in the same mapping.

A mapping can return user-defined mapping outputs or system-defined mapping outputs.

User-defined mapping outputs

A user-defined mapping output is a numeric value or date that a mapping returns by aggregating a field or expression from each row in the mapping. For example, you might need to know when orders reach a specific threshold. You can configure the mapping to return the total order amount that the mapping processed. Define a mapping output called TotalOrderAmt and configure the mapping to summarize the Order_Amount field from every row. Define an expression or port name to aggregate in an Expression transformation.
**System-defined mapping outputs**

A system-defined mapping output is a built-in value that the mapping always returns whenever the mapping completes. The mapping returns the number of source rows, the number of target rows, and the number of error rows that the mapping processes. You might pass these values in workflow variables to another task in a workflow, such as a Notification task or an Exclusive Gateway task. You do not have to define a system-defined mapping output.

Perform the following tasks with mapping outputs:

**Save the mapping output in the repository**

You can configure a Mapping task to persist a mapping output value in the repository. You can assign a persisted mapping output value to a Mapping task input. For example, you can configure the mapping to return the latest sequence number that it generated. Persist a Last_Seq_Num mapping output in the repository. The next time the mapping runs, you can use Last_Seq_Num as an the starting sequence number.

**Bind outputs to workflow variables**

You can bind mapping outputs to workflow variables and then pass the values to other tasks in a workflow. You can bind mapping outputs from the current Mapping task run to workflow variables. You can also bind persisted mapping outputs from a previous Mapping task run to workflow variables in the current run.

---

**User-Defined Mapping Outputs**

A user-defined mapping output is a numeric value or a date that a mapping returns by aggregating a field or expression from each row in the mapping. Define the expression to aggregate and the data type of the result value.

Define the mapping output in the **Outputs** tab of the Properties view. Configure a name for the mapping output, a data type for the result, and indicate what type of aggregation to perform to return a result.

After you define the mapping output name and the output type, configure an Expression transformation in the mapping. In the Expression transformation, define the output expression that you want to aggregate. The expression can contain a port name or it can contain an expression with ports, functions, and parameters.

**Outputs View**

Define the mapping outputs in the **Outputs** view of the mapping **Properties**. When you define each mapping output, enter a mapping output name, the mapping output type, and the type of aggregation to perform.

The following image shows the mapping outputs in the **Outputs** tab of the mapping **Properties** view:
The Outputs view contains the following fields:

**Name**
The name of the output. Default is Output.

**Type**
The type of the mapping output. You can select a numeric type or a date/time type. Default is Integer.

**Precision**
The length of the mapping output field. Default is 10.

**Scale**
The digits to the right of the decimal in the mapping output field. Default is zero.

**Aggregation Type**
You can choose one of the following types of aggregation:

**SUM**
Returns the sum of the field or expression from each input row that the Data Integration Service processed.

**MIN**
Returns the smallest numeric value or date that the Data Integration Service processed from a specific field or expression in each input row.

**MAX**
Returns the largest numeric value or date that the Data Integration Service processed from a specific field or expression in each input row.

**Binding**
The name of a mapplet or logical data object output to bind to the mapping output. This field is blank unless the mapping output is returned from a mapplet instead of from an Expression transformation in the mapping.

**Description**
The description of the mapping output.
Mapping Output Expression

Configure a mapping output expression in the Mapping Outputs view of an Expression transformation. The mapping output expression is a field or an expression to aggregate from the rows that the Expression transformation receives.

Configure an Expression transformation in the mapping and include the output expressions that you want to aggregate. The location of the Expression transformation in the pipeline might affect the mapping output results depending on whether the mapping contains filters or active transformations. You can add more than one Expression transformation to the mapping if you need to aggregate rows in different pipelines.

The following image shows the expressions in the Mapping Outputs view of the Expression transformation:

The Mapping Outputs view has the following fields:

**Outputs Name**

The name of a mapping output that you created at the mapping level. You must create the mapping output at the mapping level first. When you add the mapping output in the Expression transformation, you select the output name from a list of the outputs that you have already created.

**Expression**

The expression to aggregate for each row in the mapping. Enter a port name or enter an expression in the Expression Editor. The expression result must be numeric or a date. You can use parameters in the expression. The Data Integration Service applies the expression for each row that the Expression transformation receives. Each mapping output returns one value when the mapping completes.

**Note:** You do not specify the type of aggregation to perform in the Expression transformation. You indicate the field or expression that the mapping aggregates as it processes each row.
System-Defined Mapping Outputs

System-defined mapping outputs are mapping outputs that each mapping generates. You do not have to configure the aggregation for system-defined mapping outputs. You can pass system-defined mapping outputs to workflow variables.

A mapping returns the following types of system-defined mapping outputs:

- `numberOfTargetRows`
  - The number of rows that the mapping wrote to the target.

- `numberOfSourceRows`
  - The number of rows that the mapping read from the source.

- `numberOfErrorRows`
  - The number of error rows that the mapping generated.

Configure the workflow variables to assign the system-defined mapping outputs to on the **Output** tab of the Mapping task **Properties** view.

The following image shows the system-defined mapping outputs on the **Output** tab:
Persisted Mapping Outputs

You can save the mapping outputs in the repository if you run the mapping in a workflow. You can use a mapping output in a subsequent run of the same Mapping task. You can also assign persisted mapping outputs from the previous Mapping task run to workflow variables for the current Mapping task run.

Persist a mapping output in a Mapping task. A Mapping task is an instance of the mapping in a workflow with the mapping configuration and parameter bindings. For more information about Mapping tasks, see the Informatica Developer Workflow Guide.

When you click the Persistence tab in the Mapping task Properties view, the Developer tool displays all the mapping outputs for the mapping. To persist any mapping output, enable Persist for the mapping output and select the type of aggregation to perform to return a persisted value.

When you a persist mapping output, you can configure a different aggregate function for the persisted value than the aggregate function you defined at the mapping level. The Data Integration Service generates more than one mapping output value. For example, the OrderDate mapping output might contain the MIN OrderDate. The persisted OrderDate mapping output might contain the MAX OrderDate.

You can bind the mapping output from a Mapping task to the input parameter of the Mapping task the next time it runs. Feedback binding is when you configure the results from one mapping run as input to the same mapping the next time it runs. You must persist the mapping output in a Mapping task to use it for feedback binding.

An example of feedback binding is to persist the latest order date that the mapping processes. The next time the Mapping task runs, the input parameter to the mapping is the last date processed. The mapping can filter the parameter source rows to include the rows with an order date greater than the last order date processed.

The following image shows the Persistence tab on the Properties view of the Mapping task:

![Persistence tab on Properties view](image)

The Persistence tab has the following fields:

**User-Defined Output**

- The name of a mapping output that the mapping returns.

**Persist**

- Enables the Data Integration Service to persist the mapping output in the repository.
Aggregate Function
The type of aggregation to perform on the mapping output to persist. Select MIN, MAX, or SUM. The
default is the value from the mapping output that you define in the mapping properties. You can change
the aggregate function type of the persisted mapping output. You can persist a different value in the
repository than the mapping output value that you pass to workflow variables.

Description
Describes the mapping output to persist in the repository.

Persisted Values Maintenance
You can list, update, and reset the persisted mapping outputs in the repository.
You can run the following infacmd commands for persisted mapping task values:

listMappingPersistedOutputs
Lists the persisted mapping outputs and their values for a Mapping task instance in a workflow.

setMappingPersistedOutputs
Updates or resets the persisted mapping outputs for a specific Mapping task instance in a workflow.
When you reset the values, you remove the persisted values from the repository. To set mapping outputs
enter space-separated name-value pairs of mapping outputs in the command line. To reset mapping
outputs use the -reset option with a space-separated list of mapping outputs.

For more information about infacmds, see the Informatica Command Reference.

Persisted Mapping Outputs and Deployment
When you redeploy a workflow or you change a mapping output, you can affect the state of persisted
mapping outputs.
Consider the following rules and guidelines for persisted mapping outputs:

• When you deploy a workflow as an application for the first time, you do not have to perform any additional
tasks if a Mapping task has persisted mapping outputs.

• When you redeploy an application, you can choose whether to retain the state information or to discard it.
If you choose to retain the state information, the mapping output values do not change in the repository
when you redeploy the application. Otherwise the mapping outputs state is removed from persistence.

• The mapping outputs state is not backed up when you back up and restore a mapping or workflow.

• If you rename or recreate a mapping output, you cannot use the persisted mapping output value from a
previous workflow run.

Bind Mapping Outputs to Workflow Variables
After you add a mapping to a workflow, you can bind mapping outputs to workflow variables. You can pass
the values to other tasks in the workflow.

For example, you might want the Data Integration Service to evaluate the mapping output value and then
determine which object to run next. Or, you might want the Data Integration Service to use the mapping
output value in a field in the next task.
To use the mapping output in another task, bind the mapping output to a workflow variable in the Mapping task **Output** view.

**Note:** If you assign a mapping output to a workflow variable and the mapping processes no rows, the output is NULL. The mapping task does not change the value of the workflow variable. The variable retains the same value as it was before the Mapping task ran.

The following image shows the Mapping task **Output** view.

The **Output** column contains the following types of mapping outputs:

**System-Defined Mapping Outputs**

Built-in mapping outputs that the transformations return to the mapping. The system-defined mapping outputs contain the number of source rows, the number of target rows, and the number of error rows the mapping processed.

**User-Defined Mapping Outputs**

You can bind persisted mapping output values and current mapping output values to workflow variables.

**Persisted Values**

The user-defined mapping output values from the previous workflow run. The persisted value is a value that is in the repository from the last time the Mapping task ran. The persisted value is not the value that the current mapping aggregates.

**Current Values**

The user-defined mapping output values from the current Mapping task.

For more information about workflow variables, see the *Informatica Developer Workflow Guide*. 
Mapping Outputs In Mapplets

You can configure a mapplet to return mapping outputs. You can bind the mapping outputs from a mapplet to the mapping outputs at the mapping level.

When you include a mapplet in mapping, the mapplet computes the value of the outputs and passes the output values to the mapping. You can bind more than one output from a mapplet to the same output at the mapping level. You can also bind system-defined outputs from a mapplet to the mapping outputs. The mapplet outputs and the mapping outputs must be the same type.

For example, a mapplet might return the maximum value of a Salary port, a Bonus port, and a Commission port in three mapping outputs.

The following image shows the Out_Salary, Out_Bonus, and Out_Commission mapping outputs in the Outputs view:

The Outputs view contains the following fields:

- **Name**
  - The name of the output. Default is Output.

- **Type**
  - The type of the mapping output. You can select a numeric type or a date/time type. Default is Integer.

- **Precision**
  - The length of the mapping output field.

- **Scale**
  - The digits to the right of the decimal in the mapping output field.

- **Aggregation Type**
  - The type of aggregation to perform on the output expression. Choose SUM, MIN, or MAX. Default is SUM.

- **Binding**
  - The name of an output from another mapplet to bind to this mapping output. The Binding field is blank unless the mapplet contains another mapplet that is returning the mapping output.
Description

The description of the mapping output.

For each mapping output in the mapplet, create an associated output expression in the Expression transformation. Each expression identifies the fields to aggregate.

The following image shows the mapping output expressions in the Expression transformation:

For this example, the Expression transformation aggregates the Salary and Commission port values. The Out_Bonus mapping output is an expression that includes the Bonus port value plus 10% of the Bonus.

Bind Mapplet Outputs to Mapping Outputs

If a mapplet computes mapping outputs, you need to pass the output values from the mapplet to a mapping.

Bind the mapplet outputs to the mapping outputs on the mapping Outputs view.

The following image shows the Max_Compensation mapping output at the mapping level:

At the mapping level, you can bind the Salary mapplet output, the Bonus mapplet output, and the Commission mapplet output to the same mapping output called Max_Compensation.

To bind a mapplet output to a mapping output, click the Binding column for the mapping output. A list of available mapplet outputs appears. The list contains mapplet outputs that are the same type and aggregation as the mapping output. Select the mapplet outputs to assign to the mapping output.

The following image shows the Outputs Binding dialog box:
After you select the outputs to bind to Max_Compensation, the Binding field contains the following text:


The following image shows the mapping outputs in the Binding field:

Set the mapping output aggregation type is MAX. The Data Integration Service returns the highest compensation value that it finds from the Salary, the Bonus, or the Commission ports.
Mapping Outputs in Logical Data Objects

A logical data object can contain a Read or Write mapping. You can configure these mappings to return mapping outputs. You can bind the mapping outputs from the logical data object to the mapping outputs in the mapping.

When you include the logical data object in a mapping, the Read or Write mapping computes the value of the mapping outputs. The logical data object passes the output values to the mapping. You can bind more than one output from the logical data object mapping to the same output at the mapping level. You can also bind system-defined outputs from the logical data object to the mapping outputs. The logical data object mapping outputs and the mapping outputs must be the same type.

How to Configure Mapping Outputs

When you configure mapping outputs, define the mapping outputs at the mapping level, configure the expressions to aggregate at the transformation level, and persist the results at the Mapping task level.

The following image shows the process to configure mapping outputs:

To configure mapping outputs, perform the following steps:

1. Create the mapping.
2. In the Outputs view of the mapping, define the mapping output name and type of aggregation.
3. Add an Expression transformation to the mapping and configure the mapping output expression in the Expression **Mapping Outputs** view.

4. To create a Mapping task, add the mapping to a workflow.

5. Persist the mapping output in the Mapping task **Persistence** view and configure the aggregation function type for the persisted value.

6. Assign the persisted mapping output to an input parameter in the Mapping task.

7. If you want to use the mapping output in another workflow task, assign the mapping output to a workflow variable.

**Creating a Mapping**

Create a mapping that contains a reusable Filter transformation. The Filter transformation filters rows that have order dates less than a specific date. The filter expression includes a parameter called `Last_Order_Date_Parm`.

1. Create a mapping to process order data from a `Customer_Order` file.
2. In the mapping Properties view, click the **Parameters** tab.
3. Add a date/time mapping parameter called `Last_Order_Date_Parm`.
   Enter a default date for the starting parameter.
   The following image shows the mapping parameter:

4. Create a reusable Filter transformation to filter `Customer_Order` rows.
5. Define a parameter in the Filter transformation called `Order_Filter`.
   Enter a default date for the starting parameter.
6. Add a filter expression to find order dates that are greater than the parameter:
7. Add the Filter transformation to the mapping.
8. Click the Filter transformation to display the transformation Properties view.
9. Click the Parameters tab.
10. To bind the Order_Filter_Parm transformation parameter to the Last_Order_Date mapping parameter, click the Instance Value column for the Order_Filter_Parm.
11. Select Last_Order_Date.

The following image shows where the mapping parameter is bound to the transformation parameter:

---

Defining Mapping Outputs

Create a mapping and define the mapping outputs in the mapping Properties. Each mapping output definition describes what type of aggregation to perform and the data type of the results.

1. After you create a mapping, click the editor to access the mapping Properties.
2. Click the Outputs view.
3. Click New to create a mapping output.
   The Developer tool creates a mapping output with default field values.
   The following image shows the mapping output default values in the Outputs view:
4. Change the name that identifies the mapping output.

5. Select a numeric or date mapping output type. Enter the precision and scale.

6. Choose the aggregation type for the mapping output.
   
   You can summarize the output expression or you can find the minimum or maximum expression value that the mapping processed. Default is SUM.

7. Click File > Save to save the mapping output.
   
   You must save the mapping output before you can create a mapping output expression in the Expression transformation.

The following image shows a mapping output that contains the sum of a decimal field and a mapping output that contains a maximum date value:
Configuring the Mapping Output Expression

In the Expression transformation, configure the expression to aggregate for each row that the mapping processes.

1. Add an Expression transformation to the mapping.
   Consider the mapping logic before you decide where to place the transformation. The mapping output contains an aggregation of the rows that the Expression transformation receives.

2. In the Expression transformation, click the Mapping Outputs view.

3. Click **New** to add a mapping output expression.
   The Developer tool creates a mapping output with a output name that matches one of the mapping outputs you created at the mapping level. If you have more than one mapping output in the mapping Properties, select the appropriate mapping output name to use.

   The following image shows the Mapping Outputs view in the Expression transformation:

   ![Mapping Outputs View](image)

4. Click the **Expression** column to enter an expression in the Expression Editor.
   The expression can contain just a port name or it can contain functions, ports, and parameters.

   The following image shows an expression to calculate the Total_OrderAmt in the Expression Editor:

   ![Expression Editor](image)
5. Click **Validate** to verify that the expression is valid.
6. Click **OK** to save the expression.
   
   The expression appears in the **Expression** column for the mapping output.
7. Click **File > Save** to save the Expression transformation.

### Persisting Mapping Outputs

After you add the mapping to a workflow, you can persist mapping outputs from the Mapping task. You can use persisted mapping outputs as input to the Mapping task the next time it runs.

1. Add the mapping to a workflow to create a Mapping task.
2. Click the Mapping task icon in the workflow to view the Mapping task **Properties**.
3. Click the **Persistence** view.
   
   A list of the user-defined mapping outputs appears in the **Persistence** view.
4. Enable **Persist** to save the mapping output after the Mapping task runs.

5. Optionally, change the aggregation type and enter the description.

The following image shows the Persistence view for a Mapping task:

The Last_Order_Date mapping output is persisted. The aggregate function is MAX, so the Data Integration Service saves maximum order date value in the repository.
Assigning Persisted Outputs to Mapping Task Input

You can bind persisted mapping outputs from a Mapping task to the input parameters of the same Mapping task for the next time the workflow runs.

Assign the persisted latest order date value from the Mapping task as the input parameter to the same Mapping task. Configure a Filter transformation that uses a Last_OrderDate_Parm parameter to select which orders to process. The filter expression to select input rows is Order_Date > Last_OrderDate_Parm.

1. Click the Mapping task icon in the workflow to view the Mapping task Properties view.
   A list of the Mapping task input parameters and a list of the parameterized Mapping task configuration properties appears. The mapping must have a mapping parameter to assign the mapping output to.

2. Locate the mapping input parameter that you want to bind the mapping output to. Double-click the Value column to view the selection arrow.

3. Click the selection arrow to view a list of the parameters and variables that you can assign to the input parameter.

4. Scroll to the User Outputs section of the list and choose the persisted mapping output to use.
   The following image shows Last_OrderDate_Parm mapping parameter on the Mapping task Input view:

   ![Mapping task Input view with User Outputs](image)

5. Select the mapping output to assign to the parameter.
   The mapping output name appears in the value column for the input parameter.

   ![Selected mapping output](image)

6. Click File > Save to save the Mapping task.
   The Last_OrderDate_Parm is bound to the persisted order date value from the repository.
Binding Mapping Outputs to Workflow Variables

You can bind mapping outputs to workflow variables and pass the values to other tasks in the workflow.

To pass the mapping output value to another task, bind the mapping output to a workflow variable in the Mapping task Output view. You can bind mapping outputs from the current Mapping task or you can bind the persisted mapping outputs from the previous Mapping task run.

1. Add the mapping with the mapping outputs to a workflow.
2. Click the Mapping task icon in the workflow to view the Mapping task Properties.
3. In the Mapping task Properties, click the Output view.
   The Mapping task Output view shows the data that you can pass from the task into workflow variables.
4. Find the mapping output that you want to bind to a variable.
5. Double-click the Variable column to access the selection arrow and view a list of workflow variables.
   The following image shows where to bind the Total_Order_Amt mapping output to the wf_Variable_Total_OrderAmt workflow variable in the Mapping task Output view:

   ![Diagram showing binding of Total_Order_Amt to workflow variable]

6. To create a workflow variable, click the New Variable option from the list of workflow variables in the Value column.
   Enter the variable name, type, and default value.
How to Bind Mapplet Outputs to Mapping Outputs

You can configure a mapplet to return mapping outputs. You can bind the mapping outputs from the mapplet to the mapping outputs at the mapping level.

When you include a mapplet in mapping, the mapplet computes the value of the outputs and passes the output values to the mapping. You can bind more than one output from a mapplet to the same output at the mapping level. You can also bind system-defined outputs from a mapplet to the mapping outputs.

The following image shows the process to configure mapplet outputs and bind them to mapping outputs:

To bind outputs from mapplets to mapping outputs, perform the following steps:

1. Create the mapplet.
2. Define the mapplet output name and the type of aggregation in the Outputs view of the mapplet.
3. Add an Expression transformation to the mapplet and configure the mapping output expression in the Expression Mapping Outputs view.
4. Add the mapplet to a mapping.
5. Create mapping outputs in the mapping.
6. Bind the outputs from the mapplet to the mapping outputs.

Defining Mapplet Outputs

Create a mapplet and define the mapping outputs in the Outputs tab of the mapplet Properties view. Each mapping output definition describes what type of aggregation to perform and the data type of the results.

1. After you create a mapplet, click inside the mapping canvas to access the mapplet properties.
2. Click the Outputs view.
3. Click New to create a mapping output.
   The Developer tool creates a mapping output with default field values.
4. Change the name that identifies the mapping output.
5. Select a numeric or date mapping output type. If you are creating a numeric type, enter the precision and scale.
6. Choose the aggregation type for the mapping output.
   You can summarize the output expression or you can find the minimum or maximum expression value that the mapping processed. Default is SUM.
   The following image shows a mapplet output called Total_Sales_Tax with an aggregation type SUM:

7. Click File > Save to save the mapping output.
   You must save the mapping output before you can create any mapping output expression in an Expression transformation.

### Configuring a Mapping Output Expression in a Mapplet

Configure the expression to aggregate for each row that the mapplet processes.

1. Add an Expression transformation to the mapplet.
   Consider the mapplet logic before you decide where to place the transformation.
2. In the Expression transformation, click the Mapping Outputs view.
3. Click New to add an output expression.
   The Developer tool creates a mapping output with a output name that matches one of the mapping outputs you created at the mapplet level. You might have more than one output to choose from.
4. Enter an expression with the Expression Editor.
   The expression can contain a port name or it can contain functions, ports, and parameters.
5. Click Validate to verify that the expression is valid.
6. Click OK to save the expression.
   The following image shows the Mapping Outputs view with a mapping output expression that calculates a sales tax:
7. Click **File > Save** to save the Expression transformation.

### Binding Outputs from a Mapplet to Mapping Outputs

When you include the mapplet in a mapping, you can bind the outputs from the mapplet to the mapping outputs that you define at the mapping level.

1. Define a mapping and add the mapplet to the mapping.
2. Click the mapping canvas to view the mapping **Properties**.
3. Click the **Outputs** view.
4. Click **New** to create a mapping output.
   
The Developer tool creates a mapping output with default field values.
5. Change the mapping output type, the aggregation type, the precision and the scale to match fields from the mapplet that you want to bind to.
6. Optionally, change the name and enter a description.
7. Click the selection arrow in the **Binding** field to view a list of outputs.
   
The following image shows the Outputs Binding dialog box:
8. Select the mapplet output to bind to the mapping output.
   You can select more than one mapplet output to bind to the same mapping output.

9. Click OK.
   The mapplet outputs that you select appear in the Binding field.
   The following image shows the mapplet output name in the Binding field of the mapping output:
Dynamic Mappings Overview

A dynamic mapping is a mapping in which you can change sources, targets, and transformation logic at run time based on parameters and rules that you define. You can configure dynamic mappings to allow metadata changes to sources and targets. You can determine which ports a transformation receives, which ports to use in the transformation logic, and which links to establish between transformation groups.

Develop dynamic mappings to manage frequent schema or metadata changes or to reuse the mapping logic for different data sources with different schemas. When the source or target definition or metadata changes, you do not have to manually synchronize the data object and update each transformation before you can run the mapping again. When the same mapping logic applies to different sources, you do not have to develop a mapping for each source.

Dynamic mappings include the following features that you can configure:
Dynamic sources that allow changes to the metadata in flat file and relational sources at run time

When the metadata in a flat file or relational source changes, Read and Lookup transformations can get data object columns directly from the dynamic sources at run time.

Dynamic ports that can receive new or changed columns based on the data flow in a mapping

Transformations can include ports that automatically receive new or changed columns based on the data flow in a mapping. These dynamic ports receive one or more columns and automatically generate ports for each column.

Rules that can determine which columns to include or exclude in a dynamic port

You can define a set of rules to determine which columns to include or exclude in a dynamic port. These rules determine which columns propagate downstream in the mapping through a dynamic port.

Port selectors that can select input ports based on the rules that you define to use in the transformation logic

Port selectors can select one or more input ports based on the rules that you define. You can define a set of rules to determine which input ports to include or exclude. Use port selectors in a transformation logic.

Dynamic expressions that can perform repetitive calculations

Dynamic expressions can include dynamic ports or port selectors to repeat the expression for each generated port in the dynamic port or for each port in the port selector at run time.

Dynamic targets that can generate columns for flat file and relational targets at run time

Dynamic targets allow you to define the columns for flat file and relational targets at run time. Write transformations can generate columns at run time based on an associated data object or the mapping flow. Write transformations that represent relational targets can also create or replace tables at run time.

Run-time links at a group-level between transformations that can determine which ports to connect at run time

Mapping objects in a dynamic mapping can have new or changed port names between mapping runs that cannot be linked at design time. Create a run-time link, which is a group-to-group link that uses a parameter or a link policy to determine which ports to link at run time.

Parameters to change sources, targets, and transformation properties at run time

Dynamic mappings allow you to create parameters at different levels in a mapping. Use parameters to change sources and targets for flat files or relational resources at run time. You can also use parameters to change the transformation properties and the rules for ports at run time.

Dynamic Mapping Scenarios

Develop dynamic mappings when your organization has to manage frequent schema or metadata changes or to reuse the mapping logic for different data sources with different schemas.

Depending on the scenario, develop dynamic mappings as follows:

The schema or metadata of the data source changes frequently.

Relational or flat file data sources used in mapping projects might have frequent changes to the metadata or schema.

The following types of data integration projects often involve accommodating source definition changes:

- Handling or validating errors within a data integration project
Performing incremental loads for data migration

When you develop a dynamic mapping for this scenario, configure dynamic sources, dynamic ports, and run-time links or dynamic targets. Optionally, configure port selectors, dynamic expressions, and parameters for transformation properties.

If the columns or the column properties in the source change after you develop the dynamic mapping, you do not have to manually synchronize the data objects and update each transformation before you can run the mapping again.

**The same mapping logic applies to different data sources with different schemas.**

The data integration logic is the same but the mapping must be run against one or more sources and targets.

The following types of data integration projects need to run the mapping against different sources:

- Moving data to Hadoop or to an operational data store
- Creating slowly changing dimensions for data warehousing

When you develop a dynamic mapping for this scenario, configure sources and targets as parameters and configure dynamic ports. Configure port selectors, dynamic expressions, and parameters for transformation properties to reuse the mapping logic. If the metadata of the sources and targets also change, configure dynamic sources and dynamic targets.

**Dynamic Mapping Components**

Like any other mapping, a dynamic mapping contains Read and Write transformations, other transformations, ports in transformations, and links between ports. In addition, transformations in a dynamic mapping contain
dynamic ports and generated ports. A dynamic mapping can also contain run-time links between transformation groups.

The following image illustrates the components in a dynamic mapping.

1. Port
2. Dynamic port
3. Generated port
4. Link
5. Run-time link
6. Group

**Dynamic port**

A port in a transformation that can receive one or more columns from an upstream transformation. Dynamic ports can receive new or changed columns based on the metadata that passes through the mapping.

**Generated port**

A port that represents a single column within a dynamic port. The dynamic port automatically creates a generated port for each column based on the dynamic port rules.

**Link**

A link connects ports to propagate data from one transformation to another. In addition to links between ports, a dynamic mapping can contain the following links:

- A link between a group and a dynamic port to propagate data for one or more columns.
- A link between dynamic ports to propagate data for one or more columns.
- A link from a generated port to a port to propagate data for a single column.

**Run-time link**

A link between transformation groups that determines which ports to connect at run time based on a policy, a parameter, or both. A run-time link appears as a thick line between transformations. A run-time link enables the Data Integration Service to establish links at run time for the new or changed ports that cannot be linked to downstream transformations, reusable transformations, or the Write transformations at design time.

**Group**

Group is a set of ports in the mapping object. For example, **All Ports (8)** is a group in the Read transformation that contains eight ports. **All Ports (2)** group in the Sorter transformation contains two ports, a dynamic port and a port. In a dynamic mapping, the group enables to create dynamic ports and
run-time links. Drag a group to a downstream transformation to create a dynamic port. Press control and
drag a group to create a run-time link between groups.

Dynamic Sources

You can configure a flat file dynamic source or a relational dynamic source in a mapping. A dynamic source
is a source that can change at run time. You can use parameters to change the source at run time. Or, you
can choose to update the data object based on the data source.

Use one of the following tasks to configure a dynamic source:

**Parameterize the flat file properties or the relational table properties of the source.**

You can parameterize the file name and the directory name in a flat file physical data object. You can
parameterize the resource, the table owner, and the connection in a relational data object.

**Fetch metadata changes from the data source file or the data source table at run time.**

You can get new or changed source metadata at run time based on changes in the flat file or relational
data source.

**Parameterize the data object in the transformation.**

You can change the data object that is the source for the Read transformation. The data object is the
name of a physical data object in the Model repository. You can configure a parameter that determines
which data object to use at run time.

Parameterize File Properties or Table Properties

You can parameterize the flat file properties and relational table properties for a data object in the
transformation. Parameterize the flat file name and directory. Parameterize the resource, connection, and
table owner name for a relational data object.

**Parameterize the Flat File Name and Directory**

You might parameterize the flat file name and parameter if you need to run a dynamic mapping with similar
flat file sources. If you parameterize the file name or directory, you do not have to create separate physical
data objects for each flat file source.

You can parameterize the file name and the directory in a flat file physical data object. You can parameterize
the properties before you create a transformation from the data object. Configure the parameters on the
**Advanced** tab of the physical data object properties. When you create the transformation from the physical
data object, you can use mapping parameters to override the parameter default values.
The following image shows the Advanced tab of a physical data object:

![Advanced tab of a physical data object](image)

Parameterize the Relational Table Resource and Connection

You can parameterize the resource, the connection, and the table owner properties in the Read transformation. You might parameterize the resource if you need to run a mapping with different but similar tables in the same database. Or, you might need to change the connection to access a different database. You might have a unique SQL query that you need to run against multiple relational sources. If you parameterize the resource, you do not have to create separate physical data objects for each source table.

Configure the relational table parameters on the Run-Time tab of the transformation properties. You cannot parameterize these properties in the relational physical data object. When you create parameters for the properties in the Read transformation, you create mapping parameters.

By default, you create a connection type parameter for the connection. You configure a resource type parameter for the table name and a string parameter for the table owner.

The following image shows the Run-Time tab of the Read transformation:

![Run-Time tab of Read transformation](image)

Get Data Object Columns from the Data Source

You can update the ports in a Read transformation at run time based on the structure of the relational data source or the flat file data source.

You might use this method to update the run-time instance of the Read transformation when you expect small changes to the source. For example, you have a mapping that processes a source from another organization. The organization that creates the source cannot guarantee the order of the columns in the source file. When you enable the option to update the data object columns at run time, the Read transformation ports change based on the structure of the source data. The Read transformation passes the data to the downstream transformations in the dynamic mapping for processing.

Enable the option to At runtime, get data object columns from data source on the Data Object tab of the Read transformation.
The following image shows where to enable the option on the Data Object tab:

![Data Object Tab](image)

The Data Integration Service determines the structure of a relational source by the schema. It examines the schema of the resource that appears on the Run-time tab. The Data Integration Service then updates the columns in the transformation data object based on the schema.

The Data Integration Service can use different methods to determine the structure of a flat file source. The method is determined by the value of the Generate Run-Time Column Names property in the flat file physical data object.

**Note:** When you choose to update the data object columns from a data source at run time, the Data Integration Service updates a run-time instance of the Read transformation. It does not update the physical data object metadata or the Read transformation metadata in the Model repository. You cannot see the changes in the Developer tool. To update the physical data object definition in the Model repository, use the synchronize option in the Developer tool. The Developer tool reimports the physical data object metadata and changes the metadata.

For more information about setting the flat file physical data object properties, see the Informatica Developer Tool Guide.

### Parameterize the Data Object

You can parameterize the data object and change the source for the Read transformation at run time.

Parameterize the data object when you have a different physical data object in the Model repository for each data source. You might parameterize the data object if you need to configure the same transformation for a flat file or for a database table. When you parameterize the data object, you enable the transformation to use data objects that have different properties or unique SQL queries.

When you create a Read transformation from a physical data object, information about the data object appears on the Data Object tab of the transformation properties. You can click the data object name to view the physical data object definition from the Model repository.

To parameterize a data object, choose Specify by Parameter in the Data Object tab. The properties in the Data Object tab change.

To parameterize the data object, create a resource type parameter or browse for a resource parameter that you already created. The parameter default value is the name of physical data object in the Model repository. When you create a default parameter value, you select a physical data object name from a list of data objects in the repository.

When you change the data object, the transformation ports change. You can view the ports on the Ports tab in the transformation properties.

The following image shows the Data Object tab when you specify the data object by a parameter:
The Data Object tab has the following options by parameter:

**Parameter**

The name of a resource parameter that you configured as the data object. Read-only.

**Description**

The description of the parameter. Read-only.

**New**

Create a resource parameter. Browse for and select a data object in the Model repository for the parameter default value.

**Browse**

Browse for a resource parameter and select the parameter.

**Default value**

The default value of the resource parameter that you configured for the data object. The default value is a physical data object name. Read-only.

---

**Where to Configure a Dynamic Source**

You can use parameters to change the source at run time. Or, you can configure the Data Integration Service to update the data object based on the data source.

The following table lists where you can configure dynamic sources:

<table>
<thead>
<tr>
<th>Method</th>
<th>Location</th>
<th>Source Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a data object by parameter</td>
<td>Data Object tab</td>
<td>Flat file</td>
</tr>
<tr>
<td>At run time, get data object columns from data source</td>
<td>Data Object tab</td>
<td>Flat file</td>
</tr>
<tr>
<td>Assign a parameter to a flat file name and directory</td>
<td>Flat file physical data object Advanced tab.</td>
<td>Flat file</td>
</tr>
<tr>
<td>Assign a parameter to the connection, owner, or resource</td>
<td>Run-Time tab</td>
<td>Relational</td>
</tr>
</tbody>
</table>
Dynamic Ports and Generated Ports

Transformations in a dynamic mapping can include dynamic ports, which can receive new or changed columns from an upstream transformation. Dynamic ports receive one or more columns and create generated ports based on input rules. A generated port is a port within a dynamic port that represents a single column. A dynamic port fetches all columns from an upstream transformation and creates a generated port for each column. You can define input rules to filter the columns that a dynamic port receives.

Use dynamic ports in a transformation in the following cases:

**To receive new and changed columns**

When a Read transformation reads from a dynamic source or a parameterized source, create a dynamic port in the downstream transformations to receive new and changed columns.

**To filter columns based on input rules**

When you have to process only a certain type of columns in a transformation, create a dynamic port and define input rules to filter columns. For example, a mapping source has columns with decimal, string, and date/time data type. You need to process data only for columns of decimal data type. Create a dynamic port and define input rules to include only decimal columns.

**To repeat a calculation for more than one port in an Expression transformation**

When you have to perform the same calculation on more than one port, use dynamic ports in dynamic expressions. A dynamic expression runs one time for each port in the dynamic port and returns the result to a dynamic output port.

The following transformations can include dynamic ports:

- Aggregator
- Expression
- Filter
- Joiner
- Lookup
- Rank
- Router
- Sequence Generator
- Sorter
- Update Strategy

**Note:** Dynamic mappings can contain other transformations that cannot include dynamic ports. However, you might have to manually update the mapping when the source metadata changes.

You can create a dynamic port from the All Ports group of a Read transformation, a group of an upstream transformation, or a dynamic port in an upstream transformation. The Developer tool creates dynamic ports with the data type value as dynamic. You can change the dynamic port name. You cannot edit the generated ports.
The following image shows a dynamic port From_Read_Employee with generated ports:

If a Read transformation reads from a dynamic source, dynamic ports in the downstream transformations automatically fetch any new or changed columns. For example, if a new column "title" is added to the dynamic source, the Read transformation passes the new column to the dynamic port and the dynamic port creates a generated port for the "title" column.

The following image shows a new generated port "title" that the dynamic port From_Read_Employee automatically receives from the Read transformation:

A transformation can contain more than one dynamic port. A transformation can also contain ports and dynamic ports.
The following image shows an Expression transformation with two ports and two dynamic ports:

When you add ports of the same name to a transformation, the Developer tool appends a number to the dynamic port or the generated port to resolve port naming conflicts. Port names take precedence over generated port names. For example, when a transformation has a port ssn and you create a dynamic port that has a generated port of the same name, the Developer tool changes the name of the generated port in the dynamic port as ssn1.

**Note:** Any changes to the port attributes automatically propagates to the generated ports in the pipeline. You do not need to select Propagate Attributes for a dynamic or a generated port to manually propagate the changed port attributes.

**Input Rules for Dynamic Ports**

Input rules are a set of conditions in a dynamic port that determine which generated ports to create within a dynamic port. You can define input rules for a dynamic port to filter the columns that a dynamic port receives.
and to create generated ports only for the filtered columns. Use input rules to propagate specific columns downstream in the mapping.

When you have to process only a certain columns or certain type of columns in a dynamic port, define input rules to filter the columns based on the names or data type. For example, an employee database table has many columns with decimal, string, and date/time data type. You need to process data only for columns that begin with SAL and of data type decimal. Create a dynamic port and define input rules to include only the columns that meet this condition.

You can define input rules to change ports in a dynamic port in the following ways:

Include or exclude ports

Specify which ports to include or exclude in a dynamic port based on the port names or data type. You can define multiple rules. The Data Integration Service applies rules in the order in which they appear in the input rules list. Include all ports is the default input rule.

Include all remaining ports

Add ports that are excluded from other dynamic ports in the transformation. When you have multiple dynamic ports in a transformation, you can include all remaining ports from the upstream transformation in the last dynamic port.

Rename generated ports

Add a prefix or a suffix to the generated port names. Use prefixes or suffixes to indicate where ports occur in a mapping.

Reorder generated ports

Display generated ports according to the order of the rules that you enter. By default, the Developer tool displays the ports in the same order as they appear in the upstream transformation.

Use the Input Rules dialog box to define input rules, rename the generated ports, change the order of the generated ports, and view the results of the rules.

The following image shows the Input Rules dialog box with a default input rule to include all ports from an upstream transformation.
The dialog box has the following areas:

**Include or Exclude Ports area**

The Include or Exclude Ports area provides options to define one or more input rules and to include remaining ports in the last dynamic port.

**Settings area**

The Settings area provides options to rename generated ports and to reorder generated ports.

**Port Preview area**

The Port Preview area displays the generated ports based on the rules and settings that you define. You can view the results to verify the combination of rules and edit the rules.

### Include or Exclude Ports

You can include or exclude ports based on the port names or data type. Each input rule uses an operator and selection criteria to filter ports. You can define multiple rules. The Data Integration Service applies rules in the order in which they appear in the input rules list. Include all ports is the default input rule.

Configure the following input rule settings to determine which ports to include or exclude:

**Operator**

Determines whether to include or exclude ports. Default setting is to include ports.

**Selection Criteria**

Determines whether to filter ports based on port names or data types.

**Detail**

Determines which ports to filter based on the details that you provide for port names or data type.

The following table describes the selection criteria and how to specify the details for the criteria:

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Description</th>
<th>Criteria Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Includes all ports. Do not use this selection criteria with exclude operator.</td>
<td>You do not have to specify any details.</td>
</tr>
<tr>
<td>Name</td>
<td>Filters ports based on the port name.</td>
<td>Select the port names from a list of values or use a parameter of type Port or Port List.</td>
</tr>
<tr>
<td>Type</td>
<td>Filters ports based on the data type of the port.</td>
<td>Select data types from a list.</td>
</tr>
<tr>
<td>Pattern</td>
<td>Filters ports based on the pattern of the port name.</td>
<td>Choose prefix, suffix, or regular expression as the pattern type for the port name. Then, enter a value for the pattern or use a parameter of type String.</td>
</tr>
</tbody>
</table>

You provide the selection criteria details in an **Input Rule Detail** dialog box that appears based on the selection criteria. For example, you provide the Name selection criteria details in the **Input Rule Detail By Name List** dialog box.
Include All Remaining Ports

When you have multiple dynamic ports in a transformation, you can define the last dynamic port to include ports that are not part of other dynamic ports in the transformation. Choose the **Include all remaining ports** option for the last dynamic port.

For example, a source contains columns of type string and decimal. You remove the leading spaces from the string columns in a table and write the output of the string data along with the data for all other columns to the target table.

In an Expression transformation, you can configure a dynamic port String_Ports to include only the string ports that you want to transform and another dynamic port Remaining_Ports to include all the remaining ports from the Read transformation. String_Ports has generated ports of type string and Remaining_Ports has generated ports for all the remaining ports from the All Ports group of the Read transformation.

The following image shows two dynamic ports String_Ports and Remaining_Ports in the Expression transformation:

![Image of two dynamic ports](image)

Rename Generated Ports

Rename generated ports such that the port names are unique within a transformation. Add prefixes or suffixes to indicate where the generated ports occur in a mapping.

When you add ports of the same name to a transformation, the Developer tool appends a number to the generated port to resolve port naming conflicts. When you use a dynamic source and if a port name conflict occurs at run time, the Data Integration Service renames the generated port.

When generated ports are automatically renamed, the Data Integration Service fails to run the mapping in the following cases:

- An unresolved link exists from the renamed generated port to a port.
- A transformation property, such as group by port or join condition, uses the renamed generated port.

To avoid such mapping failures, rename generated ports such that the names are unique within each transformation.
Example

An Expression transformation has three dynamic ports. The "From_Read_Emp" and "From_Read_Emp2" dynamic ports include the generated port "Title." To avoid a name conflict, the Developer tool renames the generated port in "From_Read_Emp2" to "Title1."

The following image shows the renamed generated port Title1 in the Expression transformation:

You link the generated port "Title1" in the Expression transformation to a port "Title1" in the Sorter transformation. You also use "Title1" as the sort key.

The following image shows the link from the generated port in the Expression transformation to the port in the Sorter transformation:

If you add another link from the port "Title" in the Read transformation to a port "Title" in the Expression transformation, the Developer tool renames the generated ports. The generated port in the "From_Read_Emp" dynamic port is renamed to "Title1." The generated port in the "From_Read_Emp2" dynamic port is renamed to "Title2." The link to "Title1" in the Sorter transformation appears unresolved.
The following image shows the new link between the Read and Expression transformations, the generated ports that the Developer tool renamed in the Expression transformation, and the unresolved link to the Sorter transformation:

The mapping fails at run time because the generated port that is used as the sort key might not be the intended port to use.

To avoid such mapping failures, rename generated ports such that the names are unique within each transformation. For example, you want to trim the leading spaces in the string ports of the "From_Read_Emp" dynamic port. Add a suffix _trim to the generated ports. You want to find if the ports in the "From_Read_Emp2" dynamic port have null values. Add a suffix _isNull to the generated ports.

The following image shows the generated ports that you renamed in the Expression transformation:

Reorder Generated Ports

You can reorder ports to effectively view and analyze results.

An EMPLOYEE flat file source has many columns that undergoes frequent changes. You want to sort the employees by name and view the employee data in a way that the employee names appear in the first column.
followed by the city where the employees work. You also want to move the columns that are of type decimal to the end because you do not want to analyze the data for those columns.

The following image shows the dynamic port From_Read_EMPLOYEE with the original order of generated ports:

Configure three input rules. The first rule is to include the ports by name name and city, the second rule is to include all ports by type string, and the third rule is to include all ports by type decimal.

The following image shows the order of the input rules:
If you select **Reorder generated ports according to the input rule order**, the ports that qualify the first rule appears first followed by the ports that qualify the other rules.

The following image shows the reordered generated ports after you update the input rules settings.
Links in Dynamic Mappings

A dynamic mapping can contain links between two ports, between a group and a dynamic port, between two dynamic ports, or links from a generated port to a port.

The following image shows the links in a dynamic mapping:

1. Link from a group to a dynamic port
2. Link between two dynamic ports
3. Link from a generated port to a port
4. Link between two ports

Link from a group to a dynamic port

A link from a group to a dynamic port propagates data for one or more columns. A group can contain one or more ports and dynamic ports. Input rules for a dynamic port determine the generated ports that appear under the dynamic port. The default rule is to include all columns in the group as generated ports in the dynamic port of the downstream transformation.

For example, the image shows a link from the All Ports group in the Read transformation to a dynamic port "From_Read_EMPLOYEE" in the Sorter transformation. The input rule for the dynamic port "From_Read_EMPLOYEE" in the Sorter transformation is defined to include string ports.

Link between two dynamic ports

A link between two dynamic ports propagates data for one or more columns. Input rules for a dynamic port determine the generated ports that appear under the dynamic port. The default rule is to include all columns from the upstream dynamic port as generated ports in the dynamic port of the downstream transformation.

For example, the image shows a link from the dynamic port "From_Read_EMPLOYEE" in the Sorter transformation to another dynamic port "From_Read_EMPLOYEE" in the Expression transformation. The input rule for the dynamic port in the Expression transformation includes string ports and excludes the "name" port.

Link from a generated port to a port

A link from a generated port to a port propagates data for a single column.

For example, the image shows a link from the generated port "name" under the dynamic port "From_Read_EMPLOYEE" in the Sorter transformation to a port "name" in the Expression transformation.

Generated ports within a dynamic port can change when the mapping uses a dynamic source or based on input rules. If you create a link from a generated port and the generated port is no longer available, the
Developer tool shows the link to the port as an unresolved link. For example, you update the input rule for the "From_Read_EMPLOYEE" dynamic port in the Sorter transformation to exclude the "name" port. The Developer tool changes the link to an unresolved link.

The following image shows an unresolved link to the port "name" in the Expression transformation:

The Developer tool shows a warning message for unresolved links when you validate the mapping. If the generated port is available when you run the mapping, the Data Integration Service resolves the link and processes the mapping. However, if the Data Integration Service cannot resolve the link, the mapping fails. You must remove the unresolved links to successfully run the mapping. Right-click the transformation and select Clear Unresolved Links to clear any unresolved links in a transformation.

**RELATED TOPICS:**
- "Linking Ports" on page 17

---

**Port Selectors**

When a transformation has generated ports, you need to configure the transformation to run successfully when the generated ports change. You can use a port selector to determine which ports to use in a dynamic expression, a lookup condition, or a joiner condition.

A port selector is an ordered list of ports that you can reference in an expression. When the generated ports change in a dynamic mapping, the port selector can contain different ports in it.

For example, an Expression transformation has the following expression that references a generated port in a mapping:

```
Salary * 12
```

You configure the mapping to allow three different dynamic source file formats. Each source file contains a column that contains salary information. However, the column name is different in each dynamic source: Salary, Monthly_Salary, or Base_Salary. The Expression transformation multiplies salary by 12 to return a yearly salary.

You can create a port selector in the Expression transformation to accommodate the different column names. The port selector has a selection rule that accepts any port name with the suffix Salary.

Configure the expression to include the port selector name instead of the Salary column name. The expression runs successfully with any of the salary port names.
The expression has the following syntax:

\[ \text{Salary}_\text{PortSelector} * 12 \]

You can create a port selector in an Expression transformation, a Lookup transformation, or a Joiner transformation. These transformations contain expressions that can reference all the ports in the port selector.

### Selection Rules

When you configure a port selector, you define selection rules to determine which generated ports to include. The selection rules are similar to the input rules that you can configure for dynamic ports.

A port selector can include ports or generated ports. Configure a port selector on the Port Selector tab.

The following image shows the Port Selector tab:

Configure the following properties for a port selector:

**Name**

Identifies the port selector. You can create multiple port selectors in a transformation and reference them in expressions.

**Scope**

Identifies a group of ports that the port selector applies to. You must choose the scope when you create a port selector for a Joiner or a Lookup transformation. These transformations have multiple input groups. The Joiner transformation has a Master or a Detail scope. The Lookup transformation has an Import or a Lookup scope. The Expression transformation has one input group. The scope is always All Ports.

**Selection Rules**

Determines the ports to include in the port selector. When you create the selection rules, the Port Preview panel shows the ports that qualify from the current input ports. These ports might change. Configure the selection rules to accommodate ports from different sources.

You can create selection rules based on the following criteria:

**Operator**

Includes or excludes the ports that selection rules return. Default is include. You must include ports before you can exclude ports.

**Selection Criteria**

The type of selection rule you want to create. You can create a rule based on the port type or the column name. To include ports based on the column name, search for specific names or search for a pattern of characters in the name.
The values to apply to the selection criteria. If the selection criteria is by column name, configure the string or name to search for. If the selection criteria is by port type, select the port types to include.

The following table describes the selection criteria and how to specify the details for the criteria:

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Description</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Includes all ports.</td>
<td>No details required.</td>
</tr>
<tr>
<td>Name</td>
<td>Filters ports based on the port name.</td>
<td>Select the port names from a list of values or use a parameter of type Port or Port List.</td>
</tr>
<tr>
<td>Type</td>
<td>Filters ports based on the data type of each port.</td>
<td>Select data types from a list.</td>
</tr>
<tr>
<td>Pattern</td>
<td>Filters ports by a string of characters in the name or by a regular expression.</td>
<td>Choose prefix, suffix, or regular expression as the pattern type for the port name. Then, enter a value for the pattern or use a parameter of type String.</td>
</tr>
</tbody>
</table>

Where to Use Port Selectors

You can configure Port Selectors in some transformations that can process multiple fields in the transformation logic.

You can configure a port selector in the following transformations:

Expression transformation

You can reference a port selector in a dynamic expression. When you reference a port selector in the expression, the expression runs against each port in the port selector. The dynamic expression returns a result to a separate output port for each port in the port selector. If the transformation has multiple expressions that reference port selectors, the transformation returns additional output ports for each expression.

Joiner transformation

You can reference two port selectors in a join condition. Define a port selector for the master group and a port selector for the detail group. The Data Integration Service compares each port in the master group to the port in the detail group based on the order of the ports in the port selector. You can choose one type of operator to compare each pair of ports. Each port selector must have the same number of ports.

For example, you might configure a port selector called Master-SelectorX that contains the following ports: A, B, and C. You configure Detail-SelectorY that contains the following ports: D, E, F. If the join condition is Master-SelectorX = Detail-SelectorY, then the Developer tool creates the following join condition: A = D AND B = E AND C = F.

Lookup transformation

You can configure a port selector for the input ports and a port selector for the lookup ports in a lookup condition. The Data Integration Service compares each port in the input port selector to a port in the lookup port selector based on the order of the ports in each port selector. Each port selector must have the same number of ports.
Dynamic Expressions

When you configure an expression in a dynamic output port, the expression becomes a dynamic expression. A dynamic expression can generate multiple output ports.

You can reference a port selector or a dynamic port in a dynamic expression. When the port selector or dynamic port contains multiple ports, the dynamic expression runs against each port.

When you configure a dynamic expression, the Developer tool does not validate if the generated ports are valid types for the expression. For example, if you reference a port selector that contains decimal type ports in an expression that requires string types, the expression appears as valid at design time.

Example

An Expression transformation has the following generated input ports:

- EMPNO: Decimal
- NAME: String
- SALARY: Decimal
- DEPTNO: Decimal

The transformation contains a dynamic output port called MyDynamicPort. The output port returns the results of a dynamic expression. The dynamic expression multiplies the value of each port in a port selector by 100. The expression runs one time for each port in the port selector. Each instance can return a different result. The Expression transformation generates a separate output port for each result.

The Decimal_Selector port selector has a selection rule that includes the ports that are of decimal data type:

- EMPNO: Decimal
- SALARY: Decimal
- DEPTNO: Decimal

The following image shows a dynamic expression that references the Decimal_Selector port selector:
Edit the output port settings to change output port names and output port properties. You can also choose the base port.

Dynamic Targets

When you create transformations in mappings, you can configure them to change metadata, ports, or targets at design time or run time.

Mappings can contain transformations that represent targets, like flat files or relational database tables, that frequently undergo changes. For example, if a database administrator drops or adds columns, or changes column datatype and other metadata, mapping objects might be out of sync with target objects. Unless the transformation synchronizes this information at run time, the mapping fails.

Dynamic targets can automatically get metadata changes from physical data targets, including relational and flat files and customized data objects. They can also generate columns based on the upstream column definitions.

When you configure transformations to get metadata from targets, you can configure transformations to update dynamically to remain in sync with target objects.

You can configure Write transformations to access dynamic targets in the following ways:

Use customized data objects with a parameter as a target.

You can create a customized data object as a Write transformation, and specify a parameter value as the target for the transformation. With a parameter value as a target, when you change the value of the parameter, the target changes for all objects that use the parameter.

Define target columns based on mapping flow.

You can define columns for the target based on an associated data object or on mapping flow. When you create or replace the target table at run time, target columns are determined by the target object. When you define columns by mapping flow, target columns are determined by the source or by upstream transformations. For example, an upstream transformation changes the port order and metadata, and the Write transformation picks up the changes.

Enable the **Columns defined by: Mapping flow** option on the **Ports** tab.

The following image shows how the Write transformation changes to show that input is dynamic:

![Dynamic Target Configuration](image)

**Fetch metadata changes from relational targets at run time.**

You can update the structure of the data object at run time based on changes in the target.

For example, a Write_SalesTransaction in a mapping writes data to the SalesTransaction table in a relational database. If a new column is added to the SalesTransaction table after you create the
mapping, the Data Integration Service can fetch the new columns to the target table when you run the mapping.

Enable this option when you have configured an associated data object in the Data Object tab of the Write transformation.

Enable the option **At runtime, get data object columns from data source** in the Data Object tab.

**Create or replace relational target tables at run time.**

The Data Integration Service can drop the target table at run time and replace it with a table based on a target table that you identify.

Enable the option **Create or replace target table at runtime**. With this setting, the mapping performs the following tasks during run time:

1. Drop the table associated with the write object, if the table exists.
2. Recreate the table based on settings in the mapping, or create it if it did not previously exist. The Integration Service creates the table based on the associated table object or on the mapping flow.
3. Load the physical table with data.

For example, a Write_SalesTransaction in a mapping writes data to the SalesTransaction table in a relational database. If a new column is added to the SalesTransaction table after you create the mapping, the Data Integration Service can fetch the new columns to the target table when you run the mapping.

For more information about configuring dynamic targets, see "Rules and Guidelines for Dynamic Mappings" on page 121.

**Dynamic Targets Properties**

The following table lists which properties are available for Write transformations with flat file or relational sources:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Location</th>
<th>Flat File or Relational Target?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify by parameter</td>
<td>Data Object</td>
<td>Both</td>
</tr>
<tr>
<td>At run time, get data object columns from data source</td>
<td>Data Object</td>
<td>Relational</td>
</tr>
<tr>
<td>Assign a parameter value to the connection, owner, or resource</td>
<td>Run-Time tab</td>
<td>Relational</td>
</tr>
<tr>
<td>Create or replace table at run time</td>
<td>Advanced tab</td>
<td>Relational</td>
</tr>
</tbody>
</table>

**Primary Key - Foreign Key Support in Dynamic Targets**

You can configure the Write transformation to retain primary and foreign keys in relational data objects and in the relational target.

Relational data objects in a mapping can retain primary and foreign keys when you have enabled one of the following combinations of settings in the Write transformation:

- You enable mapping flow, and the ports of the Write transformation match primary and foreign key columns in the data object that the Model repository stores.
- You enable the option to get data object columns at run time, and the refreshed ports of the Write transformation match primary and foreign key columns in the data object that the Model repository stores.

The target can retain primary and foreign keys when you have enabled one of the following combinations of settings:

- You enable mapping flow and the option to create or replace the target at run time, and the ports of the Write transformation match primary and foreign key columns in the data object that the Model repository stores.
- You enable the option to get data object columns at run time, and the refreshed ports of the Write transformation match primary and foreign key columns in the data object that the Model repository stores.

**Run-time Links**

Links connect the ports between transformations to propagate data in a mapping. Mapping objects in a dynamic mapping can have new or changed port names between mapping runs. In such cases, it is not possible to link all the ports at design time. Create a run-time link, which is a group-to-group link between transformations that uses a parameter, a link policy, or both to determine which ports to link at run time. Run-time links appear as thick lines in the mapping editor.

Create a run-time link between groups of the mapping objects if the ports in the upstream transformation can change at run time. The Data Integration Service determines which ports to link at run time based on a parameter, link policy, or both that you define.

Dynamic mapping can contain both links and run-time links between mapping objects. You can manually link ports that do not change between mapping runs.

The following image shows a run-time link between the All Ports groups of an Aggregator transformation and a Write transformation:

![Run-time Link Example](image)

Use the Run-time Linking dialog box or the Run-time Linking tab in the Properties view of the downstream mapping object to create a run-time link.

Create a run-time link in the following cases:

- **The downstream transformation receives data through ports from a generated port that can change.**

  A mapping has a Read transformation that uses a parameter to change the sources or that can get metadata changes from the source at run time. The downstream transformations receive data through ports from a generated port that can change between mapping runs. Create and configure a run-time link
to the downstream transformations. At run time, the Data Integration Service connects the ports based on the link policy or the parameter values and passes the data from the generated port to the port.

**Write transformation columns are based on a data object and the ports in the Write transformation can change.**

A mapping has a Write transformation that defines columns based on an associated data object. The Write transformation uses a parameter to change the target or gets metadata changes from the target at run time. Create and configure a run-time link to the Write transformation.

**Note:** Do not create a run-time link to a Write transformation when you define the target columns based on the mapping flow.

For example, a mapping has a reusable Aggregator transformation that calculates the total salary by department. The Aggregator transformation receives employee data from an Expression transformation that has generated ports. The Expression transformation might generate the following output ports from a dynamic expression:

- **Read_EMP Source 1:**
  - EMPNO_OUT
  - NAME_OUT
  - SALARY_OUT
  - DEPTNO_OUT
- **Read_EMP Source 2:**
  - EMPNUM_OUT
  - FULLNAME_OUT
  - SALARY_OUT
  - DEPT

The Aggregator transformation instance does not have a dynamic port.

Set the run-time link properties of the Aggregator transformation instance to receive the EMPNO_OUT or EMPNUM_OUT employee number, NAME_OUT or FULLNAME_OUT string, the SALARY number, and the DEPTNO_OUT or DEPT_OUT department number.

The following image shows the links between the Expression transformation and the Aggregator transformation:

![Image showing links between transformations](image.png)

Use one of the following methods to create and configure run-time links:

**Run-time Linking dialog box**

Press Ctrl and drag the group from an upstream transformation to a group in the downstream transformation to open the **Run-time Linking** dialog box.
The following image shows the **Run-time Linking** dialog box:

![Run-time Linking dialog box](image)

**Linking dialog box**

To create a run-time link to a Write transformation, a reusable transformation, or a transformation in which you cannot create dynamic ports, you can also drag a group from an upstream transformation to a group in the reusable transformation or a Write transformation. Then, select **Create Run-time Link** in the Linking dialog box to open the **Run-time Linking** dialog box.

The following image shows the **Linking** dialog box:

![Linking dialog box](image)

**Run-time Linking tab**

Select the **Run-time Linking** tab in the **Properties** view of the downstream transformation.

The following image shows the **Run-time Linking** tab of the Write transformation:

![Run-time Linking tab of Write transformation](image)

You can perform the following tasks on the **Run-time Linking** tab or the **Run-time Linking** dialog box:
Create a run-time link.

In the Links area, click the New button and select the transformation from which you want to link the ports at run time in the New Link dialog box.

The Links area lists the group ports from which the link originates if the transformation has incoming run-time links.

Configure the run-time link properties.

In the Link Properties area, configure the run-time link properties based on a parameter, link policy, or both.

Run-time Link Properties

Configure run-time link properties to determine which ports to link between mapping objects at run time. You can define a parameter or select link policy or use both to determine which ports to link. Use the Run-time Linking dialog box or the Run-time Linking tab in the Properties view of the downstream mapping object to configure the run-time link properties.

Configure run-time links to connect ports at run time based on a parameter, a link policy, or both:

Use a parameter when the port names can change between mapping runs and you know the port name values.

Use a parameter of type Input Link Set to connect ports by name values between mapping runs. The syntax for the Input Link Set mapping parameter consists of comma-separated pairs of ports: Afield1->Bfield2, Afield3->Bfield4.

For example, you create a parameter Cust_InputLinkSet of type new Input Link Set and provide the default value as follows:

C_Name->Cust_name, C_MailID->Cust_email

When you run the mapping, the Data Integration Service creates a link between the ports C_Name and Cust_name and between C_MailID and Cust_email.

You can change the values of the parameter for the next mapping run as follows:

CustFirstName->Cust_name, CustEmail->Cust_email

Select the link policy when the ports have matching names.

Use the link policy when you want to automatically link ports that have matching names. For example, when both of the mapping objects contain a port called SALARY, the Data Integration Service links them.

You can configure the link policy to ignore a suffix or a prefix in the port names. For example, if you configure the link policy to ignore the suffix "_OUT", the Data Integration Service links SALARY_OUT to SALARY.

At run time, the Data Integration Service establishes and resolves links between the ports in the following order:

- Links that you manually create in the mapping editor.
- Links based on the parameter that you configured for a run-time link.
- Links based on the link policy that you configured for a run-time link.
A parameter is a constant value that you can change between mapping runs. In a dynamic mapping, you can use parameters to change sources and targets for flat files or relational resources. You can also use parameters to change the input rules for a dynamic port, selection rules for a port selector, transformation properties, and run-time links.

The following table lists the dynamic mapping components and properties for which you can use parameters to change values at run time:

<table>
<thead>
<tr>
<th>Dynamic Mapping Component</th>
<th>Property or Setting to Parameterize</th>
</tr>
</thead>
</table>
| **Read transformation**   | - Use a string parameter to change the input file name or directory of a flat file or relational source.  
- Use a connection parameter to change the connection of a relational source.  
- Use a resource parameter to change a flat file data object, customized data object, or a relational data object. |
| **Transformations**        | **Dynamic ports**                   |
|                           | - Use a port list parameter to change the input rule criteria details by name.  
- Use a string parameter to change the input rule criteria details by pattern. |
|                           | **Port selectors**                  |
|                           | - Use a port list parameter to change the selection rule criteria details by name.  
- Use a string parameter to change the selection rule criteria details by pattern. |
|                           | **Transformation properties**        |
|                           | - Use a port list parameter to change the group by port in an Aggregator or a Rank transformation.  
- Use an expression parameter to change the filter condition of a Filter transformation.  
- Use an expression parameter to change the join condition of a Joiner transformation.  
- Use an expression parameter to change the lookup condition of a Lookup transformation.  
- Use a sort list parameter to change the sort key in a Sorter transformation. |
| **Run-time links**         | - Use an input link set parameter to change the set of ports to link between transformation groups. |
| **Write transformation**   | - Use a string parameter to change the output file name or directory of a flat file or relational target.  
- Use a connection parameter to change the connection of a relational target.  
- Use a resource parameter to change a flat file data object, customized data object, or a relational data object. |
Related Topics:

- "Mapping Parameters Overview" on page 35

Rules and Guidelines for Dynamic Mappings

Use the following rules and guidelines when you develop and run dynamic mappings.

General Rules and Guidelines for Dynamic Mappings

- Mapplets are supported as components of a dynamic mapping.

Dynamic Sources

- Flat files and relational resources are supported as dynamic sources.

Dynamic Ports and Input Rules

Each transformation that uses dynamic ports might have unique rules and guidelines.

For more information, see the Informatica Developer Transformation Guide.

The following rules and guidelines apply to all transformations that use dynamic ports and input rules:

- When you use the New button to create a new port or dynamic port, the Developer tool assigns a default name. Rename these ports and dynamic ports such that the port names are unique within each transformation.
  
  For example, change the name of the dynamic port from Field to Dynamic_DecimalPorts.

- Create at least one include input rule for a dynamic port.

- Do not create an "exclude all" input rule that excludes all ports from a dynamic port.

- Name list and pattern values are not case sensitive.

- Rename the generated ports such that the port names are unique within each transformation.

- Create input rules in the order in which you want the Data Integration Service to evaluate the rules.

- You can search for dynamic ports but you cannot search for generated ports from the Search dialog box.

Dynamic Targets

General Guidelines for Dynamic Targets

Use the following guidelines when you configure a Write transformation to write to dynamic targets:

- If you want to retain the indexes and permissions for the target table, do not enable the option to replace the target table at run time.

- Enable automatic synchronization with the input ports of the relational target only when the associated data object defines the target columns.

- You can configure the Write transformation to define target columns by mapping flow and also to create or replace the target table at run time. These options do not conflict, but they determine target columns differently. If you configure the Write transformation to create or replace target tables at run time, the target object determines the columns of the target table. However, if you also enable the option to define target columns based on mapping flow, the target table is created or replaced based on the columns from the mapping flow.

- If you enable the option to create or replace the target table at run time based on the mapping flow, the Data Integration Service creates the target table with all of the columns in the Write transformation.
However, the Data Integration Service writes data only to the ports that are linked to the Write transformation. Unlinked columns remain empty in the target table.

- Configure the Write transformation to get metadata directly from the target at run time only when the associated data object defines the target columns. Do not configure the Write transformation to get metadata from the target at run time with the following options:
  - Define target columns by mapping flow
  - Synchronize input ports when column metadata changes

**Defining Target Columns by Mapping Flow**

Use the following guidelines when you configure a dynamic target to define its columns by upstream objects in the mapping:

- You can choose how ports are propagated to the target. When you define target columns by mapping flow, the mapping propagates all ports from the source or from upstream objects. If the mapping has a requirement for propagating concrete ports of certain types, or with certain names, then use a run-time linking policy in the **Run-time Linking** tab.

**Defining Target Columns by the Associated Data Object**

Use the following guidelines when you configure a Write transformation to create or replace the target table based on the associated data object:

- If you use a customized data object as the target and you enable the option to create or replace the target table at run time, then the name of the table that the mapping creates at run time will be the same as the table that the customized data object connection points to.
  
  For example, a mapping has a Read transaction that uses connection A; a Write transaction that uses connection B; and a customized data object that uses connection C. The name of the created table is the same as the table that connection C points to.
  
  **Note:** While each connection can point to different database instances, all the connections in a dynamic mapping must all be of the same database type. For example, all the connections must be Oracle connections. Otherwise, a validation error results.

- If the target table has a cyclic dependency among other tables in the database, do not configure the Write transformation with the option to create or replace the target table at run time. When a cyclic dependency exists, the database cannot execute the command from the Data Integration Service to drop or create a table, so the mapping fails.

- When you configure your mapping with the option to create or replace the target table at run time, the Data Integration Service creates a table, and never a synonym or view, regardless of the type of the relational resource in the target object.

- The data types in the target table might be different from the data types in the Write transformation. When the Data Integration Service runs the mapping, it might change data types between upstream transformations and the target table.

**Run-time Links**

- If the mapping flow defines the target columns of the Write transformation, do not create a run-time link to the Write transformation.

**Links in Virtual Table Mappings and Operation Mappings**

- In a virtual table mapping, you cannot create a run-time link or a dynamic port and cannot link a generated port to a port for virtual tables.

- In an operation mapping, you cannot create a run-time link or a dynamic port and cannot link a generated port to a port for Fault, Input, and Output transformations.
Developing and Running Dynamic Mappings

Develop a dynamic mapping to manage changes to source metadata and to reuse the data integration logic for different sources and targets. Run the dynamic mapping for same or different sources and targets that might have metadata changes.

The following table lists the high-level tasks to develop and run a dynamic mapping. The tasks and the order in which you perform the tasks depend on the mapping scenario and the transformations that you plan to use in the mapping.

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a mapping and add mapping objects.</td>
<td>“Creating a Mapping” on page 16</td>
</tr>
<tr>
<td></td>
<td>“Adding Objects to a Mapping” on page 16</td>
</tr>
<tr>
<td>Configure dynamic sources for the Read or Lookup transformations to get metadata changes from flat file or relational sources at run time.</td>
<td>“Configuring a Dynamic Source” on page 124</td>
</tr>
<tr>
<td>Create dynamic ports in transformations and link ports.</td>
<td>“Creating a Dynamic Port” on page 125</td>
</tr>
<tr>
<td>Define input rules for dynamic ports to determine which generated ports to create.</td>
<td>“Configuring Dynamic Ports Using Input Rules” on page 126</td>
</tr>
<tr>
<td>- Define input rules to include or exclude ports.</td>
<td></td>
</tr>
<tr>
<td>- Rename the generated ports.</td>
<td></td>
</tr>
<tr>
<td>- Optionally, reorder generated ports.</td>
<td></td>
</tr>
<tr>
<td>Configure the transformations.</td>
<td>Refer to the Informatica Developer Transformation Guide for details to configure transformations in the mapping.</td>
</tr>
<tr>
<td>Optionally, create port selectors to use in the transformation logic of Joiner, Lookup, or Expression transformations.</td>
<td>“Creating a Port Selector” on page 130</td>
</tr>
<tr>
<td>Optionally, create dynamic expressions to use in Expression transformations.</td>
<td>“Creating a Dynamic Expression” on page 131</td>
</tr>
<tr>
<td>Configure Write transformations to write to dynamic targets as follows:</td>
<td>“Configuring a Dynamic Target” on page 133</td>
</tr>
<tr>
<td>- Define column definitions from an associated data object and get metadata changes from the target file or define columns definitions from the mapping flow of the upstream transformation.</td>
<td></td>
</tr>
<tr>
<td>- Create or replace target tables at run time for Write transformations that represent relational targets.</td>
<td></td>
</tr>
<tr>
<td>Create and configure a run-time link to determine which ports to link at run time.</td>
<td>“Creating and Configuring a Run-time Link” on page 136</td>
</tr>
<tr>
<td>After you determine where you want to use parameters in the mapping, create and assign parameters.</td>
<td>“How to Configure Parameters” on page 56</td>
</tr>
<tr>
<td>- Configure sources as parameters</td>
<td></td>
</tr>
<tr>
<td>- Configure targets as parameters</td>
<td></td>
</tr>
<tr>
<td>- Configure transformation properties as parameters</td>
<td></td>
</tr>
<tr>
<td>Validate, compile, and run the dynamic mapping.</td>
<td>“Validating and Running a Dynamic Mapping” on page 138</td>
</tr>
</tbody>
</table>
Configuring a Dynamic Source

You can configure the Read and Lookup transformations in a mapping to dynamically update metadata, including column names, when transformation sources change.

When you configure Read and Lookup transformations for a dynamic mapping, you can use one or more of the following methods:

Use a parameter as a source

When you use a parameter value as a source for a Read or Lookup transformation, you choose a parameter name that references a source data object that you defined elsewhere in the repository.

Configure Read and Lookup transformations to get metadata at run time

When you configure Read or Lookup transformations to get data object columns from a source at run time, the mapping refreshes the port definitions when the mapping runs.

Using a Parameter as a Source for a Dynamic Mapping

You can use a parameter as a source for a dynamic mapping source object.

1. In the mapping editor, select the source object.
2. In the Properties view, click the Data Object tab.
3. To use different values for the source object between mapping runs, select Parameter in the Specify by list.
4. Click New to create a new parameter, or Browse to choose an existing parameter.

Configuring Sources to Get Metadata Changes at Run Time

You can configure data sources for source objects in mappings to get metadata changes at run time.

When the metadata for data source columns changes after you develop a mapping, the mapping might be out of date. You can configure data sources with an option to get this data at when the mapping runs.

1. In the mapping editor, select the source object.
2. In the Properties view, click the Data Object tab.
3. To dynamically get columns from the data source file at run time, select **At run time, get data object columns from the data source**.

![Image with dynamic port configuration](image)

**Creating a Dynamic Port**

Create a dynamic port to receive multiple columns from an upstream transformation. The columns can change at run time. You can create more than one dynamic port in a transformation.

1. Create a dynamic port in the following ways:
   - Drag the All Ports group or a dynamic port from another transformation.
     - The Developer tool creates a dynamic port with generated ports for all columns in the upstream transformation, and links the ports. You can change the input rules to filter the generated ports.
     - The following image shows the dynamic ports in the Sorter and Expression transformations:
   - In the **Properties** view of the transformation, select **New Dynamic Port** on the **Ports** tab.
     - The Developer tool creates an empty dynamic port that you can configure. You must manually link the ports to create generated ports.
The following image shows the new dynamic port in the Sorter transformation with no generated ports:

![Image of Sorter transformation](image)

2. Optionally, you can change the name of the dynamic port and add a description of the port. The generated ports inherit port properties from the upstream transformation, and are not editable.

## Configuring Dynamic Ports Using Input Rules

Define input rules to specify which ports to generate and propagate through a dynamic port in the pipeline. Use the **Input Rules** dialog box to define input rules for a dynamic port, rename the generated ports to indicate where ports occur in a mapping, change the order of the generated ports, and view the results for the rules. You can add multiple rules to include and exclude ports. The Data Integration Service applies rules in the order that they appear in the list.

1. Open the Input Rules dialog box.
2. Define one or more input rules for each dynamic port in the transformation. For each input rule, perform the following steps:
   a. Choose the operator and the selection criteria for the input rule.
   b. If you choose the Name selection criteria, specify the criteria details by name or by parameter.
   c. If you choose the Type selection criteria, select the data type of the ports from the list.
   d. If you choose the Pattern selection criteria, select the pattern type, and specify the pattern string as value or as parameter.

   Optionally, define the input rule for the last dynamic port in the transformation to include all remaining ports from the upstream transformation.

3. Rename the generated ports.
4. Optionally, reorder the generated ports.
5. Verify the results of the input rules and the input rule settings.

### Step 1. Open the Input Rules Dialog box

Open the **Input Rules** dialog box to define or edit input rules.

- Open the **Input Rules** dialog box in the following ways:
Right-click the dynamic port in a transformation, and select **Edit Input Rules**.

On the **Ports** tab of the transformation, click **Input Rules** for the dynamic port.

The **Input Rules** dialog box appears with a default Include All input rule.

**Step 2. Define Input Rules**

Define input rules to include or exclude ports that a dynamic port receives from the upstream transformation.

1. In the **Include or Exclude Ports** area, choose **Input Rules**.
2. To edit the default input rule, perform the following steps:
   a. Choose the operator and selection criteria for the input rule.
   b. Configure the selection criteria details.
3. Optionally, add more input rules in the order that you want the Developer tool to run the rules.
   a. Click **New** to add a new row for the input rule.
   b. For each input rule, choose the operator and the selection criteria, and specify the criteria details.
4. To include only the remaining ports from the upstream transformation, perform the following steps:
   a. Create another dynamic port or choose the last dynamic in the transformation.
   b. Choose **Include all remaining ports**.

   This rule includes ports from the upstream transformation that are not part of other dynamic ports.

**Step 2a. Choose the Operator and Selection Criteria**

Choose an operator to include or exclude ports and a selection criteria to filter ports based on port names or data types.

1. From the **Operator** column, choose the **Include** or **Exclude** operator.
   The operator determines whether the rule must include or exclude ports.

   ![Include/Exclude Ports](image)

2. From the **Selection Criteria** column, choose one of the following options:
   - **All**. Includes all ports. Do not choose this option with the Exclude operator.
   - **Name**. Includes or excludes ports based on port names.
   - **Type**. Includes or excludes ports based on data type of the port.
• **Pattern.** Includes or excludes ports based on patterns of port names.

3. From the **Details** column, click the Details arrow to provide the selection criteria details. The **Input Rule Detail** dialog box for the selection criteria appears.

**Step 2b. Configure the Name Selection Criteria Details**

If you choose the name selection criteria for the input rule, select the port names from a list of values. Or, use a parameter of type port or port list to specify port names that you can change at run time.

1. In the **Input Rule Detail: By Name** dialog box, select one of the following from the **Specify by** list:
   • **Value.** Enter port names, or select port names from a list.
   • **Parameter.** Create a new parameter or choose an existing parameter of Port List type.

2. Specify port names values in one of the following ways:
   • Enter the port names in the **Names** box and click **Add**.
   • Click **Choose**, select the port names in the **Ports** dialog box, and click **OK**.

3. To create a new parameter for the port name, perform the following steps:
   a. Click **New**.
   b. In the **Parameters** dialog box, enter a parameter name.
   c. Optionally, enter a parameter description.
   d. Enter a default value for the port name parameter. You can also click **Choose to select port names from the list of ports**.
   e. Click **OK**.

4. To choose an existing parameter for the port name, perform the following steps:
   a. Click **Browse**.
   b. In the **Assign Parameter** dialog box, choose a parameter.
   c. Optionally, create new parameters or edit parameters from this dialog box.
   d. Click **OK**.

**Step 2c. Configure the Type Selection Criteria Details**

If you choose the type selection criteria for the input rule, select the types from a list of data types.

1. In the **Input Rule Detail: By Type** dialog box, select the data types from the list.

2. Click **OK**.
Step 2d. Configure the Pattern Selection Criteria Details

If you choose the pattern selection criteria for the input rule, choose a pattern type for the port name. Enter a value for the pattern or use a parameter of type String to specify the value that you can change at run time.

1. In the Input Rule Detail: By Pattern dialog box, select one of the following from the Pattern Type list:
   - Prefix. To include or exclude port names that start with the prefix string.
     For example, if you enter the prefix value E, the input rule filters the port names that start with E or e such as EmpNo, empName, and EmpTitle.
   - Suffix. To include or exclude port names that end with the suffix string.
     For example, if you enter the suffix value E, the input rule filters the port names that end with E or e such as empname and EMPTITLE.
   - Regular expression. To include or exclude port names that follow a certain pattern.
     For example, if you enter the value E.*No, the input rule filters port names that start with E and end with No such as ENo, EmpNo, and EmployeeNo.

2. Select one of the following from the Specify by list:
   - Value. Enter the string value for the pattern.
   - Parameter. Create a new parameter or choose an existing parameter of string type.

3. Specify pattern values in the String box and click OK.

4. To create a new parameter for the pattern, perform the following steps:
   a. Click New.
   b. In the Parameters dialog box, enter a parameter name.
   c. Optionally, enter a parameter description.
   d. Enter a default value for the pattern and enter the precision value.
   e. Click OK.

5. To choose an existing parameter for the port name, perform the following steps:
   a. Click Browse.
   b. In the Assign Parameter dialog box, choose a parameter.
   c. Optionally, create new parameters or edit parameters from this dialog box.
   d. Click OK.

Step 3. Rename the Generated Ports

Rename generated ports to make sure that the port names are unique within a transformation.

1. In the Settings area, choose Rename Ports.
2. Choose whether to add a prefix or a suffix to rename the generated ports.
3. Add the text to prefix or suffix the generated ports.

The renamed ports appear in the Port Preview area.
Step 4. Reorder the Generated Ports

Reorder generated ports to effectively view and analyze results.

- In the Settings area, choose Reorder generated ports according to the input rule order.
  
The reordered ports appear in the Port Preview area. The generated ports appear according to the order of the input rules instead of the order in which it appears in the upstream transformation.

Step 5. Verify the Dynamic Port Configuration

View the generated ports based on the rules and settings that you defined for the dynamic port.

- In the Port Preview area of the Input Rules dialog box, verify the results of the input rule settings for the dynamic port.

Creating a Port Selector

Create a port selector to determine which ports to use in a dynamic expression, a lookup condition, or a joiner condition.

1. Click the Port Selectors tab.
2. In the Port Selectors area, click New.
   
The Developer tool creates a port selector with a default selection rule that includes all ports.
3. In the Port Selectors area, change the port selector name to a unique name.
4. If you are working on the Joiner transformation or the Lookup transformation, choose the scope.
   
The available ports change based on the group of ports that you choose.
5. In the Selection Rules area, select an Operator.
   - Include. Create a rule that includes ports for the port selector. You must include ports before you can exclude ports.
   - Exclude. Create a rule that excludes specific ports from the port selector.
6. Choose the Selection Criteria.
   - By Name. Select specific ports by name. You can select the port names from a list of ports in the scope.
   - By Type. Select ports by type. You can select one or more data types.
   - By Pattern. Select ports by a pattern of characters in the port name. You can search with specific characters or you can create a regular expression.

The following image shows the Port Selector tab:

![Port Selector Tab](image)

7. Click the Detail column.
   
The Input Rule Detail dialog box appears.
8. Select the values to filter ports by.
   - By Name. Choose to create a port list by value or by a parameter. Click Choose to select the ports in the list.
   - By Type. Select one or more data types from a list. The Port Preview area shows ports of the types that you select.
   - By Pattern. Choose to search the prefix or suffix of the port name for a specific pattern of characters. Or, choose to create a regular expression to search with. Configure a parameter or configure the pattern to search with.

The Port Preview area shows the ports in the port selector as you configure the rules.

9. To reorder the ports in the port selector, select Reorder generated ports according to the input rule order.

**Creating a Dynamic Expression**

Create a dynamic expression in an Expression transformation to run the expression one time for each port in a dynamic port or a port selector. The dynamic expression returns the results to a separate generated port for each instance.

1. In the Expression transformation, go to the Properties view and click the Ports tab.
2. Click New Dynamic Port.
   The Developer tool creates a dynamic port with default properties.
3. Rename the dynamic port and disable the input option.
   The dynamic port must be an output port.
4. In the Expression column for the dynamic output port, click the Open button ( ).
   The Dynamic Expression dialog box appears:

5. In the Expression editor, enter an expression. The expression can include a port selector or a dynamic port.
For example, `LTRIM(RTRIM(Dynamic_Customer))`, where `Dynamic_Customer` is a dynamic port.

6. Click **Validate** to validate the expression.

7. Click **OK** to exit the **Validate Expression** dialog box.

8. In the **Output Port Settings** area, select the dynamic output port from the **Base Port** list or choose a port selector that you referenced in the expression.

   The Developer tool generates output ports based on what you select.

9. Use the following steps to rename the output ports:
   - Click **Edit Output Port Settings**.

   The **Output Port Settings** dialog box appears.

   ![Output Port Settings dialog box]

   a. In the **Name** list, select one of the options and enter a value for the prefix or suffix. If you selected **Fixed string + Auto-number**, enter the text for output port name. For example, if you enter TRIM for the output port name, the output port names appear as `TRIM1`, `TRIM2`, `TRIM3`.

   b. Optionally, choose **Specify settings** in the **Other Settings** area to change the type, precision, and scale for the output ports. By default, the output ports use the settings of the base ports.

   c. Click **OK**.

10. Click **OK** to exit the **Dynamic Expression** editor.
Configuring a Dynamic Target

You configure a Write transformation to receive columns from the target at run time when target metadata changes. Optionally, specify a parameter as the target data object to enable the assignment of different values. You can also specify whether the Write transformation uses an associated object or a mapping flow for port definitions.

When you configure the Write transformation for a dynamic mapping, you can use one or more of the following methods:

**Use a parameter as a target**

Specify a parameter as the underlying data object for the target to enable you to change the schema for the Write transformation through the parameter.

**Get data object columns from the target at run time**

Enable the option to get data object columns from the target at run time to dynamically update Write transformation ports with changes in the target schema.

**Define Write transformation ports from mapping flow**

When you choose to define ports from mapping flow, the Data Integration Service defines Write transformation ports based on upstream column definitions. Target columns update dynamically at run time.

Using a Parameter as a Target for a Dynamic Mapping

You can use a parameter as the data object for the transformation and then change the parameter at run time.

1. Select the Write transformation in the mapping editor.
2. In the Properties view, click the Data Object tab.
3. Select Parameter in the Specify by list.
4. Select one of the following options:
   - Click **New** to create a parameter. Name the parameter, and then browse to select a default value for the parameter.
   - Click **Browse** to select an existing parameter.
The following image shows a transformation with a parameter as a data source:

Getting Target Object Columns from the Data Source at Run-Time

You can enable the option to get data object columns from the data source at run-time.

When you select the option to get data object columns from the data source at run time, the mapping fetches the data object columns to the transformation when the mapping runs. If the data source columns and metadata have changed, then the mapping fetches the changed information.

1. In the Properties view, click the Data Object tab.
2. Select At run time, get data object columns from the data source.

Defining Write Transformation Ports

Define target object columns by mapping flow to enable upstream mapping objects to update the incoming ports for the Write transformation.

1. In the Properties view, click the Ports tab.
2. Select Columns defined by: Mapping flow.
3. Enable dynamic ports and targets:
   a. Drag upstream ports to the input pane of the Write transformation. The target gets column definitions from upstream mapping objects.
   b. In the Properties view, click the Advanced tab.
   c. Select Create or replace table at run-time.

The following image shows the Create or replace table at run-time option in the Advanced tab of the target object:

At run-time, the Data Integration Service creates, or drops and replaces, the target table.

**Note:** When a mapping contains multiple targets whose columns are defined by the same physical data object, enable the Create and replace table at run-time option for only one of the targets. If you enable this option for more than one target, the metadata of the table that the mapping creates would match only one of the targets, and the mapping fails.
Creating and Configuring a Run-time Link

Create a run-time link between transformation groups to link ports at run time based on a parameter or a link policy or both.

1. Create a run-time link in the following ways:
   - Press Ctrl and drag the group to a downstream transformation in the dynamic mapping. The Run-time Linking dialog box appears.
   - To create a run-time link to a Write transformation or a reusable transformation, drag a group from an upstream transformation to a group in the reusable transformation or a Write transformation. Then, select Create Run-time Link in the Linking dialog box to open the Run-time Linking dialog box.
   - In the downstream transformation to which you want to create a run-time link, go to the Properties view and click the Run-time Linking tab.

2. In the Link Properties area, select one or both from the following options to determine which ports to link at run time:
   - Parameter. Use parameter if the port names can change between mapping runs and you know the port names. You can create a new parameter or choose an existing parameter of type Input Link Set.
   - Link Policy. Use link policy to automatically link ports by name. This option is selected by default. If the port names contain a prefix or suffix, enter the string to ignore.

3. To create a new parameter of Input Link Set type, perform the following steps:
   a. Click New.
   b. In the Parameters dialog box, enter a parameter name. For example, Cust_InputLinkSet.
   c. Optionally, enter a parameter description.
d. Enter a default value for the parameter as comma-separated pairs of ports.
   For example, enter the default value as follows:
   C_NAME->Cust_name, C_ACCTBAL->Cust_acctbal

e. Click OK.

4. To choose an existing parameter of Input Link Set type, perform the following steps:
   a. Click **Browse**.
   b. In the **Assign Parameter** dialog box, choose a parameter.
   c. Optionally, create new parameters or edit parameters from this dialog box.
   d. Click **OK**.

5. Optionally, to add another run-time link from the **Run-time Linking** dialog box, perform the following steps:
   a. Click the **New** button ( ) in the **Links** area.
      The **New Link** dialog box appears.
      
      a.[Image](image-url)  
      b. Choose a group from another transformation in the dynamic mapping.

6. Click **OK** to create a run-time link.
The Developer tool creates run-time links between the groups.

7. To edit an existing run-time link, right-click the link, and select **Run-time Linking**.

   The **Run-time Linking** dialog box appears where you can change the options to determine which ports to link.

**Validating and Running a Dynamic Mapping**

Validate a mapping to ensure that the Data Integration Service can read and process the entire mapping, and run the mapping to write the transformed data to the target.

1. Open the mapping, click **Edit > Validate**.
   If errors appear in the **Validation Log** view, fix the errors and validate the mapping again.
2. When the mapping is valid, click **File > Save** to save the mapping.
3. Click **Run > Run Mapping**.
   The **Run Mapping** window displays the progress of the mapping run. The mapping runs and writes the output to the target file.
4. Click **Window > Show View > Progress** to view the progress of the mapping run.
   The **Progress** view opens.
5. Change parameter values between mapping runs.
6. Validate and re-run the mapping after the source schema has changed or after you changed the parameter values.

**Troubleshooting Dynamic Mappings**

Consider the following troubleshooting tips when you design and test dynamic mappings:

*My dynamic mapping failed with run-time errors for parameters and links. I would like to make sure that the parameters and links successfully resolve before I run the mapping.*

When you run a dynamic mapping, the Data Integration Service compiles the mapping to complete the following tasks:

- Resolve parameter values.
- Expand dynamic ports and convert generated ports to static ports.
- Link static ports.
- Resolve run-time links to connect the ports.

You can view the optimized mapping to see the compiled version of a mapping. Right-click an empty area in the mapping editor and click **Show Optimized Mapping**. The Data Integration Service generates the optimized mapping. You can review the optimized mapping, fix any issues, and then run the mapping.

The following image shows a dynamic mapping that contains transformations with dynamic ports:
The metadata in the flat file or relational source, or relational target does not match the metadata of the Read, Write, or Lookup transformation.

When you create or replace target data at run time or when you get columns from a data source at run time, a metadata mismatch might occur. In such cases, an “invalid identifier” error appears, and the Data Viewer view might not show any results. To see mapping results, run the mapping.

After I enable dynamic targets, I get an error when I try to preview data for the target data object.

If you enable the Create or replace target at run time and Columns defined by mapping flow options, an error appears when you attempt to run data preview for the Write transformation after the completion of a mapping run. The error happens because the Data Integration Service does not synchronize metadata between the Write transformation and the target database. To use data preview, manually synchronize the Write transformation with the target database before previewing data.

My mapping failed to run when I enabled the options “At run time, get data object columns from data source” and “Enable bulk load” on the Write transformation.

In a Write transformation with a data source that contains a column with a datatype of long, you enabled the options At run time, get data object columns from data source and Enable bulk load. The mapping fails to run unless the column with the long datatype is the last column in the target table.
I used DD_UPDATE as the update expression in an Update Strategy transformation, and enabled mapping flow on the Write transformation, but my mapping fails.

You enabled the Columns defined by mapping flow option in the Write transformation. When the mapping uses DD_UPDATE as the update expression in an Update Strategy transformation, the primary key is not propagated through the mapping. As a result, the mapping fails.

To resolve the issue, choose one of the following methods:

1. Define the Write transformation based on the associated data object, instead of mapping flow. Also, configure run-time linking in the Properties view of the downstream transformation or target.
2. Define the Write transformation based on mapping flow, and designate one of the Write transformation ports as the primary key.

I lost data when I tried to replicate data from a source table to a target table, and ran the dynamic mapping from the Developer tool.

You can configure a limit on the number of characters a mapping can read. To configure this setting, select Window > Preferences, then Informatica > Run Configurations > Mapping, then click the Sources tab.

The following image shows the Sources tab in the Preferences dialog box:

If you enable the Create or replace target at run time and Columns defined by mapping flow options, and the Columns > Read up to how many characters setting is lower than the precision of one or more columns in the source table, you might see a loss of data in the target table when you run the mapping in the Developer tool.

When you enable the Create or replace target at run time and Columns defined by mapping flow options, the data type of columns in the created table might vary depending on whether you configure the Columns property to read all characters or read up to a specific number of characters. Also, when you run the mapping in a deployed application, truncation does not occur.
A dynamic mapping with a custom SQL query fails at run time.
The mapping fails at run time if you add a custom SQL query in the Query tab of the Read or Lookup transformation, and also enable the option to get data object columns from the data source at run time.

The Data Integration Service queries the physical data object or customized data object, and the results of the query contain column metadata from specific columns. When the metadata from the query results conflicts with the refreshed metadata from data source columns, the mapping fails.

The target data object contains a foreign key, but after the dynamic mapping created the table at runtime, the foreign key is missing.

When you enable the dynamic mapping option to Create or replace table at run time, the Data Integration Service might create the table without the foreign key relationship that exists in the target data object. More than one possible reason exists for this effect. Examine the mapping run log to get more information about the missing foreign key.
Dynamic Mapping Use Cases

This chapter includes the following topics:

- Use Case: Dynamic Mapping for Metadata Changes in Relational Sources, 143
- Use Case: Reuse Dynamic Mapping for Different Sources and Targets, 155

Use Case: Dynamic Mapping for Metadata Changes in Relational Sources

This section covers a use case scenario to develop a dynamic mapping that can handle metadata changes in relational sources.

Scenario

An organization receives customer data and customer order data as two tables from different departments on a weekly basis. The departments often change the order of the columns or add new columns to the tables. The organization must aggregate the total customer orders. You need a mapping to accommodate the changing source schema.

Solution

Develop a dynamic mapping that handles changes to the source schema and metadata at run time.

Configure the following dynamic mapping features:

- Read transformations that can read from dynamic sources
- Dynamic ports in the downstream transformations that can pass new and changed columns
- Write transformation that can write to dynamic targets
- Run-time links that can connect ports to the Write transformation at run time

When you run the mapping, the Data Integration Service performs the following tasks:

- Fetches the structure of the data objects and metadata changes in the source files
- Passes the new and changed columns to each transformation through dynamic ports
- Processes the data based on the transformation logic
- Connects the new and changed ports to the Write transformation
- Writes the transformed data to the target
Mapping objects

The following image shows the objects in the mapping:

The mapping contains the following objects:

**Read_CUSTOMER**
Read transformation that represents the relational source CUSTOMER. The relational table contains a separate row for each customer.

**Read_ORDERS**
Read transformation that represents the relational source ORDERS. The relational table that contains a separate row for each customer order.

**Join_CustomerOrders**
Joiner transformation that joins the CUSTOMER and ORDERS sources.

**Agg_CustomerOrders**
Aggregator transformation that aggregates the total customer orders.

**Write_CUSTOMERSUMMARY**
Write transformation that represents the relational target CUSTOMERSUMMARY. The relational table contains a column for the mapping to write the aggregated value for total orders grouped by customer.

Source tables

CUSTOMER and ORDERS are the source tables for the Read transformations in the mapping.

The following table lists the columns and metadata for the CUSTOMER table with the C_CUSTKEY column as the primary key:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_CUSTKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_NAME</td>
<td>varchar2</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>Name</td>
<td>Native Type</td>
<td>Precision</td>
<td>Scale</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>C_ADDRESS</td>
<td>varchar2</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>C_NATIONKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_PHONE</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>C_ACCTBAL</td>
<td>number(p,s)</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>C_MKTSEGMENT</td>
<td>varchar2</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The following table lists the columns and metadata for the ORDERS table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_ORDERKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>O_CUSTKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>O_ORDERSTATUS</td>
<td>varchar2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O_TOTALPRICE</td>
<td>number(p,s)</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>O_ORDERDATE</td>
<td>date</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>O_ORDERPRIORITY</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>O_CLERK</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>O_SHIPPRIORITY</td>
<td>number(p,s)</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>

Target table

CUSTOMERSUMMARY is the target table for the Write transformation in the mapping.

The following table lists the columns and metadata for the CUSTOMERSUMMARY table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_CUSTKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_NAME</td>
<td>varchar2</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>C_ADDRESS</td>
<td>varchar2</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>C_NATIONKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_PHONE</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>C_ACCTBAL</td>
<td>number(p,s)</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>
Developing a Dynamic Mapping for Metadata Changes in Relational Sources

In this example procedure, you develop a dynamic mapping that manages metadata changes in relational sources at run time.

Create a mapping `m_CustomerLoad` to configure the dynamic mapping features.

1. Add Read transformations and configure the transformations to fetch column and metadata changes directly from the dynamic sources at run time.
2. Add a Joiner transformation to the mapping, configure ports, and define the transformation logic.
3. Add an Aggregator transformation to the mapping, configure ports, and define the transformation logic.
4. Add a Write transformation to the mapping and configure the Write transformation to fetch column names from a dynamic target at run time.
5. Create a run-time link from the Aggregator transformation to the Write transformation to connect ports at run time.
6. Validate and run the mapping.

**Step 1. Configure the Read Transformations**

Configure the Read transformations to fetch column and metadata changes directly from the dynamic sources at run time.

1. Add two Read transformations that represent the CUSTOMER and ORDERS relational data objects.
2. Configure the Read_CUSTOMER transformation to fetch column and metadata changes directly from the sources at run time.
   a. Select the Read_CUSTOMER transformation.
   b. In the **Properties** view, click the **Data Object** tab.
c. Select **At run time, get data object columns from the data source.**

The following image shows the Data Object tab settings of the Read_CUSTOMER transformation:

![Data Object tab settings](image)

3. Configure the Read_ORDERS transformation to fetch column and metadata changes directly from the sources at run time.
   a. Select the Read_ORDERS transformation.
   b. In the **Properties** view, click the **Data Object** tab.
   c. Select **At run time, get data object columns from the data source.**

**Step 2. Configure the Joiner Transformation**

Add a Joiner transformation to the mapping and configure dynamic ports to receive any new and changed columns from the Read transformation. Define a join condition to join the two source tables CUSTOMER and ORDERS.

1. **Add a Joiner transformation** Join_CustomerOrders to the mapping.
2. **Create dynamic ports in the Joiner transformation:**
   a. From the Read_Customer transformation, drag the All Ports group to the Master group in the Joiner transformation.
      The Developer tool creates a dynamic port From_Read_CUSTOMER in the Master group and the Output group.
   b. From the Read_Orders transformation, drag the All Ports group to the Detail group in the Joiner transformation.
      The Developer tool creates a dynamic port From_Read_ORDERS in the Detail group and the Output group.

The dynamic ports include all ports from the corresponding Read transformations as generated ports.
The following image shows the All Ports groups from the Read transformations linked to the two dynamic ports in the Joiner transformation:

3. In the Properties view, click the Join tab.
4. Click the New button, and define the join condition as C_CUSTKEY = O_CUSTKEY_DTL.

The following image shows the Join tab with the join condition defined:

Step 3. Configure the Aggregator Transformation

Add an Aggregator transformation to the mapping and configure dynamic ports to receive any new and changed columns from the Joiner transformation. Create an aggregate expression to calculate the total price of customer orders and group the aggregation by customer.

1. Add an Aggregator transformation Agg_CustomerOrders to the mapping.
2. Create dynamic ports in the Aggregator transformation:
   a. From the Output group in the Joiner transformation, drag the From_Read_CUSTOMER dynamic port to the Aggregator transformation.
      A dynamic port From_Read_CUSTOMER appears in the Aggregator transformation.
   b. From the From_Read_ORDERS dynamic port of the Output group in the Joiner transformation, drag the O_TOTALPRICE_DTL generated port to the Aggregator transformation.
The following image shows the ports from the Joiner transformation linked to the Aggregator transformation:

3. In the Properties view, click the Ports tab.
4. Click the New button to create a port to aggregate prices of the orders.

The Developer tool creates a new port called Field.

5. Select the new port, and change the column values as follows:
   - Name: O_TOTALAMOUNT
   - Type: decimal
   - Precision: 10
   - Scale: 2
   - Input: Clear the selection to make this port an output-only port.

The following image shows the ports in the Aggregator transformation:
6. In the Expression column for the O_TOTALAMOUNT port, click the Open button.

The Aggregator Expression window appears.

7. Replace the existing expression in the editor with the following expression: \( \text{SUM(O_TOTALPRICE_DTL)} \)

8. Click Validate to validate the expression.

9. Click OK.

10. Click OK to exit the Aggregator Expression editor.

11. In the Properties view, click the Group By tab.

12. Specify the group by port to aggregate the total price by market segment as follows:
   a. Make sure that Value from the Specify by list is selected.
   b. Click Choose.
      
      The Ports dialog box appears.
c. Select the checkbox next to C_CUSTKEY and click OK.

The following image shows the selected group by port:

You can preview the Aggregator transformation data to make sure that it contains the expected results. In the mapping editor, right-click the Aggregator transformation and select Run Data Viewer. The data calculated by the transformation appears in the Data Viewer view.

Step 4. Configure the Write Transformation

Add a Write transformation and configure the Write transformation to fetch column changes directly from the target at run time.

1. Add the CUSTOMERSUMMARY relational data object as the Write transformation.
   The Write transformation appears in the editor as Write_CUSTSUMMARY.
2. Verify that the Write transformation is configured to automatically re-import metadata changes.
   a. In the Properties view, click the General tab.
   b. Make sure that Synchronize input ports is chosen.
3. Configure the Write transformation to get columns directly from the target table at run time.
   a. In the Properties view, click the Data Object tab.
   b. Select At run time, get data object columns from the data source.
Step 5. Create and Configure a Run-time Link

Create a run-time link to the Write transformation and configure a link policy to establish and resolve links by port names at run time.

1. Press Ctrl and drag the All Ports group from the Aggregator transformation to the All Ports group of the Write transformation.

   The Run-time Linking dialog box appears.

2. Verify that Link Policy in the Link Properties area is selected to automatically link ports by name at run time.

3. Click OK.

   The Developer tool creates a run-time link between the Aggregator transformation and the Write transformation.

Step 6. Validate and Run the Mapping

Validate and run the mapping. Preview the data in the target data object to verify the result.

1. In the mapping editor, click Edit > Validate.

2. When the mapping is valid, click File > Save to save the mapping.

3. Click Run > Mapping.

   The Run Mapping window displays the progress of the mapping run. The mapping runs and writes the output to the target file.
4. In the **Object Explorer** view, locate the **CUSTOMERSUMMARY** data object in your project and double click the data object.

   The data object opens in the editor.

5. Click **Window > Show View > Data Viewer**.

   The **Data Viewer** view appears.

6. In the **Data Viewer** view, click **Run**.

   The **Data Viewer** view runs and displays the data.

   In this example, the C_TOTALAMOUNT column displays the aggregated total price of customer orders.

---

**Running the Dynamic Mapping after Changes to the Source Schema**

The departments that provide the customer data table and customer order data table add a new column Comments to the tables. View the column changes in the dynamic mapping and validate and rerun the mapping. You can preview the data in the target data object to verify the updated result.

The following table lists the columns and metadata for the updated CUSTOMER table with the new C_COMMENT column:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_CUSTKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_NAME</td>
<td>varchar2</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>C_ADDRESS</td>
<td>varchar2</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>C_NATIONKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>C_PHONE</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>C_ACCTBAL</td>
<td>number(p,s)</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>C_MKTSEGMENT</td>
<td>varchar2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>C_COMMENT</td>
<td>varchar2</td>
<td>117</td>
<td>0</td>
</tr>
</tbody>
</table>
The following table lists the columns and metadata for the updated ORDERS table with the new O_COMMENT column:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_ORDERKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>O_CUSTKEY</td>
<td>number(p,s)</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>O_ORDERSTATUS</td>
<td>varchar2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O_TOTALPRICE</td>
<td>number(p,s)</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>O_ORDERDATE</td>
<td>date</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>O_ORDERPRIORITY</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>O_CLERK</td>
<td>varchar2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>O_SHIPPRIORITY</td>
<td>number(p,s)</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>O_COMMENT</td>
<td>varchar2</td>
<td>79</td>
<td>0</td>
</tr>
</tbody>
</table>

1. In the mapping editor, view the changes to the mapping.
   The Read and Write transformations automatically reflect the new columns. The dynamic ports in the Joiner and Aggregator transformations automatically have the new columns C_COMMENT and O_COMMENT from the respective Read transformations.

2. To validate the changed mapping, click Edit > Validate.
3. When the mapping is valid, click File > Save to save the mapping.
4. Click Run > Mapping.
   The Run Mapping window displays the progress of the mapping run. The mapping runs and writes the output to the target file.
5. In the Object Explorer view, locate the CUSTOMERSUMMARY data object in your project and double click the data object.
The data object opens in the editor.

6. Click **Window > Show View > Data Viewer**.

The **Data Viewer** view appears.

7. In the **Data Viewer** view, click **Run**.

The **Data Viewer** view runs and displays the data.

8. Verify that the mapping shows the expected results after the source schema has changed.

The **C_TOTALAMOUNT** column displays the aggregated total price of customer orders.

---

### Use Case: Reuse Dynamic Mapping for Different Sources and Targets

This section covers a use case scenario to develop a dynamic mapping that can remove blanks from the beginning and end of strings from different sources and write the output to different targets.

**Scenario**

An organization must clean different data files to remove beginning and end blanks in string values. The data files have different column names and have multiple columns of type string. You need a mapping that can use the same mapping logic against different sources and write to different targets.

**Solution**

Develop a dynamic mapping that can reuse the mapping logic for different sources and write to different targets.

Configure the following dynamic mapping features:

- Read transformation that uses a parameter for the data object to read from different sources
- Dynamic ports in the downstream transformations that can pass new or changed columns
- Expression transformation that contains a dynamic expression to remove leading and trailing spaces in strings
- Write transformation that creates target columns based on the mapping flow and uses a parameter in the target data object for the target file name

When you run the mapping, the Data Integration Service performs the following tasks:

- Reads the data from the appropriate source file based on the parameter value for the source data object
- Passes the new and changed columns to the Expression transformation through dynamic ports
- Expands the dynamic expression and processes the expression function for each generated port in the dynamic port
- Creates columns in the Write transformation based on the mapping flow
- Writes the transformed data to the appropriate target file based on the parameter value
Mapping objects

The following image shows the objects in the mapping:

The mapping contains the following objects:

**Read_Customer_FF**
Read transformation that represents a flat file source. The flat file contains string data with leading and ending spaces.

**Exp_TRIM**
Expression transformation that contains a dynamic expression to remove leading and trailing spaces for ports of type string.

**Exp_Output**
Expression transformation that contains transformed string ports, and remaining ports from the source object.

**Write_customerTrim_FF**
Write transformation that represents a flat file target. The mapping writes the output to the flat file target.

**Source files**
The source files are flat files that contain string data with blanks in the beginning and end. Customer_FF, orders_FF, and so on are the source files for the Read transformation.

Example procedure reads from the Customer_FF file in the first mapping run and the orders_FF file in the second mapping run.

**Customer_FF columns and data**
Customer_FF contains the following columns:

- C_Id
- C_Fullname
- C_title
- C_comp
- C_addr
- C_suite
- C_city
- C_state
- C_zip5
- C_country
- C_phone
- C_fax
- C_date
- C_email
- C_description

where the data type of C_ID and C_zip5 columns is number, and the data type of other columns is string.
Customer_FF contains the following data:

<table>
<thead>
<tr>
<th>C_id,</th>
<th>C_Fullname,</th>
<th>C_title,</th>
<th>C_comp,</th>
<th>C_addr,</th>
<th>C_city,</th>
<th>C_state,</th>
<th>C_zip,</th>
<th>C_country,</th>
<th>C_phone,</th>
<th>C_fax,</th>
<th>C_date,</th>
<th>C_email,</th>
<th>C_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Smith John, Account Executive, DMR MANAGEMENT COMPANY INC, 100 High Street, 5406, Anytown, TN, 22342, USA, 4047668150, 2124031386, 31/08/1985, <a href="mailto:bwilliams@yahoo.com">bwilliams@yahoo.com</a>, ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Balasubramanian Krishna, Account Executive, EASTON &amp; COMPANY, 71 Congress Parkway, 789, Bangalore, Karnataka, 38103, India, 4046345228, 4151689756, 29/10/1985, <a href="mailto:bmatthewc@univ.edu">bmatthewc@univ.edu</a>, ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Johnson Lara, Regional Sales Exec, GREATER BAY BANCORP, 123 Snow St., 43543, St. Paul, MN, 55103, USA, 4046581534, 6122945948, 7/9/1992, ehpuniv.edu, INACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Franklin Roosevelt, Sales Representative, JAYD TRADING, 1511 Wacker Dr, 6334, Chicago, IL, 60606, USA, 7703965851, 2065075486, 20/10/1982, <a href="mailto:trichard@univ.edu">trichard@univ.edu</a>, INACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Cruz Emilio, Sales Representative, JEFFERSON-PILOT LIFE INSURANCE, 700 Ponce de Leon Blvd, 757, Miami, FL, 33134, USA, 4043500799, 2127655499, 31/07/1983, <a href="mailto:ahelle@mailcity.com">ahelle@mailcity.com</a>, ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>King BB, Sales Representative, KUWAIT PETROLEUM CORPORATION, 18 Beale St, 967, Memphis, TN, 38103, USA, 4046243979, 2151717120, 27/09/1999, <a href="mailto:glizziem@univ.edu">glizziem@univ.edu</a>, INACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Presley Elvis, Sales Representative, PRINCIPAL PARTNERS, 45 N Green St., 43546, Tupelo, MS, 38804, USA, 4043735126, 3311313591, 1/27/1992, ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Olson Floyd, Acct MGR, SOLITON ASSOCIATES INC., 21 Lake Harriet Pkwy, 869790, Minneapolis, MN, 55410, USA, 7706425402, 3232429056, 27/08/1993, INACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Chu Steven, Account Executive, WQXR, 2100 Sepulveda Blvd, 3434, Los Angeles, CA, 90049, USA, 4042319005, 2126509756, 29/09/1988, <a href="mailto:akennetha@univ.edu">akennetha@univ.edu</a>, ACTIVE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

orders_FF columns and data

orders_FF contains the following columns:

<table>
<thead>
<tr>
<th>OrderID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
</tr>
<tr>
<td>CompanyAddress</td>
</tr>
<tr>
<td>CompanyCity</td>
</tr>
<tr>
<td>CompanyState</td>
</tr>
<tr>
<td>CompanyZip</td>
</tr>
<tr>
<td>OrderContact</td>
</tr>
<tr>
<td>DeliveryAddress</td>
</tr>
<tr>
<td>DeliveryCity</td>
</tr>
<tr>
<td>DeliveryState</td>
</tr>
<tr>
<td>PaymentType</td>
</tr>
<tr>
<td>PaymentTerms</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>DeliveryOption</td>
</tr>
<tr>
<td>DeliveryVendor</td>
</tr>
<tr>
<td>ConfirmationCode</td>
</tr>
<tr>
<td>OrderAmount</td>
</tr>
<tr>
<td>OrderType</td>
</tr>
<tr>
<td>ProductDescription</td>
</tr>
</tbody>
</table>

where the data type of Customer_ID column is number, and the data type of other columns is string.

orders_FF contains the following data:

O-5079,10110085, JOSEPHTHAL LYON & ROSS, 96 FISHER ROAD, MAHWAH, NJ, 7430, PARKE PERSLEY OR RAYFORD LECROY, 96 FISHER ROAD, MAHWAH, NJ, American Express, CHARGE, Account Executive, USA, United Parcel Service Air, 44162, $21.00, Generic, O/L/B P/W L/S TAWNY SHIMMER .08 OZ.
O-6658, 10110086, NWCA, 10255 W. HIGGINS RD., ROSEMONT, IL, 60018-5607, ROLANDA SORTO, 10255 W. HIGGINS RD., ROSEMONT, IL, American Express, CHARGE, Account Executive, USA, United Parcel Service Air, 44164, $15.64, Generic, O/L/B P/W L/S TAWNY LASTING PERFECTION .08 OZ.
O-8195, 10110087, POND EQUITIES, 4522 FT. HAMILTON PKWY., BROOKLYN, NY, 11219, CONSTANTIN PEDDICORD, 4522 FT. HAMILTON PKWY., BROOKLYN, NY, American Express, CHARGE, Account Executive, USA, United Parcel Service Air, 44164, $78.00, Generic, O/L/B P/W L/S TAWNY SHIMMER LASTING PERFECTION LIPSTYLO TAWNY SHIMMER .08 OZ.
O-9130, 10110088, SCHRODER & COMPANY, 787 SEVENTH AVENUE., NEW YORK, NY,
Add a Read transformation and configure the source data object as a parameter. Validate and run the mapping.

Add an Expression transformation

Add a Write transformation and configure the Write transformation to write to a data object that uses a file.

The target file is a flat file where the mapping writes the data after removing the blanks in the beginning and end of string values. Create a customerTrim.csv file as the target file for the target data object.

Use parameters to change the output file name at run time when you use a different data source. The Data Integration Service creates the output file based on the parameter value for the target file name and saves the file to the target directory on the system where Informatica services is installed.

Developing a Dynamic Mapping to Reuse for Different Source and Target Objects

In this example procedure, you develop a dynamic mapping to reuse the mapping logic and to run the mapping against different sources.

Create a mapping m_Replication_Template to configure the dynamic mapping features.

1. Add a Read transformation and configure the source data object as a parameter.
2. Add an Expression transformation Exp_TRIN to the mapping and configure dynamic ports.
3. Create a dynamic expression in the Expression transformation to remove leading and ending spaces in the string data.
4. Add another Expression transformation Exp_Output to the mapping and configure dynamic ports.
5. Add a Write transformation and configure the Write transformation to write to a data object that uses a parameter for the output file.
6. Configure the Write transformation to create target columns based on the mapping flow of the upstream transformation
7. Validate and run the mapping.
Step 1. Configure the Read Transformation to Use Parameters

Configure the Read transformation to use a parameter of type Resource to change the source data object between mapping runs.

1. Add a Read transformation that represents the Customer_FF flat file data object.
   The Read transformation appears in the editor as Read_Customer_FF.
2. In the Properties view, click the Data Object tab.
3. Select Parameter in the Specify by list.
4. Click New to create a new parameter.
   The Parameters dialog box appears.
5. Enter the parameter name as Param_Src.
6. Click Browse in the Default Value.
7. In the Select location dialog box, select the data object that you want to provide as the default value.
   An example default value is MRS//Cust_Department/Customer_FF, where MRS is the Model Repository Service and Cust_Department is the project where the Customer_FF data object is stored. You can change the value of the parameter when you run the mapping.

The following image shows the Data Object tab after you define the settings:

![Data Object tab after defining settings](image)

Step 2. Configure the Exp_TRIM Expression Transformation

Add an Expression transformation Exp_TRIM to the mapping, configure a dynamic port to receive any new and changed columns from the Read transformation, and create another dynamic output port to define a dynamic expression.

1. Add an Expression transformation Exp_TRIM to the mapping.
2. Drag the All Ports group from the Read_Customer_FF transformation to the All Ports group in the Exp_TRIM transformation.
   The Developer tool creates a dynamic port From_Read_CUSTOMER_FF in the Exp_TRIM transformation.
The following image shows the dynamic port in the Exp_TRIM transformation that includes all ports from the Read transformation as generated ports:

3. Define input rules to include only the string ports in the From_Read_CUSTOMER_FF dynamic port.
   a. Right-click the dynamic port, and select **Edit Input Rules**.
      The **Input Rules** dialog box appears.
   b. Select **by Type** from the **Selection Criteria** column.
   c. Click the **Details** button to select the data type you want to include.
   d. In the **Input Rule Detail: By Type** dialog box, select **string** data type from the list.
   e. Verify the **Port Preview** area to ensure that only string ports appear.
The following image shows the updated input rule and the string ports in the **Port Preview** area of the **Input Rule** dialog box:

![Input Rule Dialog Box](image)

4. Create another dynamic port as an output-only port in the **Exp_TRIM** transformation to define a dynamic expression.
   a. In the **Properties** view, click the **Ports** tab.
   b. Click **New Dynamic Port**.

   ![New Dynamic Port](image)

   The Developer tool creates an empty dynamic port Field.

   c. Clear the **Input** column to make this port an output-only port.
   d. Rename the dynamic port that you created as **Dynamic_TRIM**.

   ![Dynamic Port](image)

**Step 3. Create a Dynamic Expression in the **Exp_TRIM** Transformation**

Create a dynamic expression in the **Exp_TRIM** transformation to remove leading and ending spaces in the string data.

1. Select the **Exp_TRIM** transformation.
2. In the Properties view, click the Ports tab.

3. In the Expression column for the Dynamic_TRIM dynamic port, click the Open button (تانغ). The Dynamic Expression window opens.

4. Replace the existing expression in the editor with the following expression:

   \[
   \text{LTRIM(RTRIM(From\_Read\_Customer\_FF))}
   \]

5. Click Validate to validate the expression.

6. Click OK to exit the Validate Expression dialog box.

7. Rename the output ports for the expression as follows:
   a. In the Output Port Settings area, select the Base Port as From_Read_Customer_FF.
   b. Click Edit Output Port Settings.
      The Output Port Settings dialog box appears.
   c. In the Name list, select Base port name + Suffix.
   d. In the Suffix box, enter _OUT.
   e. Click OK.
8. Click **OK** to exit the **Dynamic Expression** editor.

The following image shows the **Dynamic_TRIM** dynamic port with the renamed generated ports:

![Image showing Dynamic_TRIM dynamic port with renamed ports](image)

**Step 4. Configure the Exp_Output Expression Transformation**

Add another Expression transformation to pass the output of the transformed strings and other data types from the Read transformation. Create a dynamic port to get the output ports from the Exp_TRIM transformation and another dynamic port to get the unused ports from the Read transformation.

1. Create dynamic ports in the Exp_Output transformation:
   
   a. From the **Exp_TRIM** transformation, drag the **DynamicTrim** dynamic port to the All Ports group in the **Exp_Output** transformation.

   The Developer tool creates a dynamic port **DynamicTrim** in the **Exp_Output** transformation.
b. From the Read_Customer_FF transformation, drag the All Ports group to the All Ports group in the Exp_Output transformation.

The Developer tool creates a dynamic port From_Read_CUSTOMER_FF in the Exp_Output transformation.

The following image shows the two dynamic ports in the Exp_Output transformation.

2. To include the unused ports in the From_Read_CUSTOMER_FF dynamic port, define the input rule as follows:

   a. Right-click the dynamic port, and select Edit Input Rules.

      The Input Rules dialog box appears.

   b. Click the New icon to add an input rule.

   c. Select Exclude in the Operator column.

   d. Select Type in the Selection Criteria column.

   e. Click the Details arrow to select the data type you want to include.

   f. In the Input Rule Detail: By Type dialog box, select string data type from the list.

   g. Verify the Port Preview area to ensure that string ports do not appear.
The following image shows the updated input rule and the ports in the Port Preview area of the Input Rule dialog box:

![Input Rule Dialog Box](image1.png)

Step 5. Configure the Write Transformation to Use Parameters

Create a customerTrim_FF data object to add as the Write transformation in the mapping and configure the data object to use a parameter of type String for the output file name.

1. Create a customerTrim_FF data object based on the customerTrim.csv file.

2. To use a parameter for the output file, perform the following steps:
   a. In the Parameters tab of the data object, click the New button ( ) to create a new parameter.
   b. In the Name column, change the parameter name as Param_TgtFile.
   c. In the Default Value column, click the Open button ( ).
      The Edit Parameter Value window appears.
   d. Enter the default file name value as customerTrim.csv and click OK.

3. Save the customerTrim_FF data object.

The following image shows the Parameter tab with the new parameter:

![Parameter Tab](image2.png)
Step 6. Configure the Write transformation to Use the Mapping Flow for Target Columns

Add the Write transformation to the mapping and configure the Write_customerTrim_FF transformation to create target files at run time based on the columns from the Exp_Output transformation.

1. Add the customerTrim_FF data object as the Write transformation to the mapping.
2. In the Properties view of the Write transformation, click the Ports tab.
3. Choose the Mapping flow option to define the columns for the target.

   The Developer tool creates a dynamic port input in the Write_customerTrim_FF transformation.

   The following image shows the Ports tab after you choose the option:

   ![Ports Tab](image1)

   The following image shows the new dynamic port input in the Write_customerTrim_FF transformation:

   ![Write_customerTrim_FF](image2)
4. From the **Exp_Output** transformation, drag the All Ports group to the input port in the **Write_customerTrim_FF** transformation.

The Developer tool creates a link and flows the columns from the All Ports group of the Exp_Output transformation to the input dynamic port of the Write transformation.

The following image shows the m_Replication_Template mapping with the Write transformation configured:

![Mapping Diagram](image)

**Step 7. Validate and Run the Mapping**

Validate and run the m_ReplicationTemplate mapping with the default parameter values for the source data object and target file to view the result.

1. In the mapping editor, click **Edit > Validate**.
2. When the mapping is valid, click **File > Save** to save the mapping.
3. Click **Run > Mapping**.

The **Run Mapping** window displays the progress of the mapping run. The mapping runs and writes the output to the target file.
4. Click **Window > Show View > Progress** to view the progress of the mapping run.

The **Progress** view appears.

![Progress View](image)

5. Click the hyperlinked text that shows the result of the mapping run.
6. Click **Go to Log**.

The mapping log opens.
7. To save the log to a file, click **File > Save a Copy As** and choose a directory.
8. To view the results written to the target file, navigate to the target directory on the system where Informatica services is installed:

   \<Informatica Services Installation Directory\>\tomcat\bin\target

9. Open the `customerTrim.csv` file to verify that the string values do not have spaces in the beginning and end.

   Each line of the file lists data for the columns in the order that they appeared in the target object as C_Id, C_zip5, C_Fullname, C_title, C_comp, and so on. For example, the first five lines of the file contain the following data where the blanks from the beginning and end of strings are removed:

   1,22342,Smith John,Account Executive,DKR MANAGEMENT COMPANY INC,100 High Street, 5406,Anytown, TN, USA, 4047668150, 2124031386, 31/08/1985,bwilliams@yahoo.com,ACTIVE
   2,38103,Balasubramanian Krishna,Account Executive, EASTON & COMPANY, 71 Congress Parkway, 789, Bangalore, Karnataka, India, 4046345228, 4151689756, 29/10/1985,bmatthews@univ.edu,ACTIVE
   3,55103,Johnson Lars, Regional Sales Exec, GREATER BAY BANCORP, 123 Snow St., 42543, St. Paul, MN, USA, 4046581536, 6122945948, 7/9/1992, ehpuniv.edu,INACTIVE
   4,94116,Zogby Kevin, Regional Sales Exec, HEWLETT-PACKARD, 317 29th St., 5856, San Francisco, CA, USA, 404262730, 4155466814, 7/8/1985, grobertwu.edu,ACTIVE
   5,60606, Franklin Roosevelt, Sales Representative, JAYD TRADING, 1511 Wacker Dr, 6394, Chicago, IL, USA, 7703965851, 2065075486, 20/10/1982, trihard@univ.edu, INACTIVE

Running the Dynamic Mapping Against Different Sources and Targets

Run the mapping for different sources.

1. Change the parameter value for the source data object and the target output file.

2. Rerun the mapping for a different source and target.

Step 1. Change the Parameter Values for the Source Data Object and the Target File

Change the parameter values for the source data object and the output file name of the target data object.

1. To change the parameter value for the source data object, perform the following steps:
   a. In the Properties view of the mapping, click the Parameters tab.
   b. Locate the Param_Src parameter for the source object.
   c. In the Default Value column, click the Open button (3).
      The Select Location dialog box appears.
   d. Select the orders_FF data object.

The following image shows the Parameter tab of the mapping with the updated default value:
The following image shows the mapping that reflects the ports from the orders_FF data object for the Read transformation. The dynamic ports reflect new generated ports.

2. To change the parameter value of the target file name, perform the following steps:
   a. Open the customerTrim_FF target data object.
   b. In the Parameters tab of the data object, locate the Param_TgtFile parameter for the target file name.
   c. In the Default Value column, click the Open button.
      The Edit Parameter Value window appears.
   d. Change the default file name value to ordersTrim.csv and click OK.

The following image shows the Parameter tab of the customerTrim_FF data object with the updated default value:

Step 2. Validate and Run the Dynamic Mapping with Different Parameter Values

Validate and run the m_ReplicationTemplate mapping for a different source and target and view the result.

1. In the mapping editor, click Edit > Validate.
2. When the mapping is valid, click File > Save to save the mapping.
3. Click Run > Mapping.
   The Run Mapping window displays the progress of the mapping run. The mapping runs and writes the output to the target file.
4. Click **Window > Show View > Progress** to view the progress of the mapping run. The **Progress** view appears.

5. Click the hyperlinked text that shows the result of the mapping run.

6. Click **Go to Log**.

   The mapping log opens.

7. To save the log to a file, click **File > Save a Copy As** and choose a directory.

8. To view the results written to the target file, navigate to the target directory on the system where Informatica services is installed:

   `<Informatica Services Installation Directory>\tomcat\bin\target`

9. Open the **ordersTrim.csv** file to verify that the string values do not have spaces in the beginning and end.

Each line of the file lists data for the columns in the order that they appeared in the target object as Customer_Id, Order_ID, Company, CompanyAddress, CompanyCity, and so on. For example, the first five lines of the file contain the following data where the blanks from the beginning and end of strings are removed:

```
10110085,0-5079,JOSEPHTHAL LYON & ROSS,96 FISHER ROAD,MAHWAH,NJ,7430,PARKE PERSLEY OR RAYFORD LECROY,96 FISHER ROAD,MAHWAH,NJ,American Express,CHARGE,Account Executive,UPA,United Parcel Service Air,44162,$21.00,Generic,0/L/B P/W L/S TAWNY SHIMMER .08 OZ.
10110086,0-6658,HRCA,10255 W.HIGGINS RD.,ROSEMONT,IL,60018-5607,ROLANDA SORTO,10255 W.HIGGINS RD.,ROSEMONT,IL,American Express,CHARGE,Account Executive,UPA,United Parcel Service Air,44163,$56.40,Generic,0-L-B PW LIPSTYLO LASTING PERFECTION .08 OZ. 10110087,0-8195,POND EQUITIES,4522 FT. HAMILTON PKWY.,BROOKLYN,NY,11219,KONSTANTIN PEDDICORD,4522 FT. HAMILTON PKWY.,BROOKLYN,NY,American Express,CHARGE,Account Executive,UPA,United Parcel Service Air,44164,$78.00,Generic,0/L/B P/W L/S TAWNY SHIMMER LASTING PERFECTION LIPSTYLO TAWNY SHIMMER .08 OZ.
10110088,0-9130,SCHRODER & COMPANY,787 SEVENTH AVENUE,NEW YORK,NY,10019,GIORGIATWITCHELL,787 SEVENTH AVENUE,NEW YORK,NY,American Express,CHARGE,Account Executive,UPA,United Parcel Service Air,44165,$14.00,Generic,A/COL L PERFECTION L/S REF P SUPREME LASTING PERFECTION LIPSTYLO TAWNY SHIMMER .08 OZ.
10110089,0-9352,YUASA TRADING COMPANY (AMERICA),150 EAST 52ND STREET,NEW YORK,NY,10005,STEIFF MCGLOWN,150 EAST 52ND STREET,NEW YORK,NY,American Express,CHARGE,Account Executive,UPA,United Parcel Service Air,44166,$54.00,Generic,O/L/B L PERFECTION REF LIPSTYLO COFFEE PEACH SUPREME .08 OZ.
```
This chapter includes the following topics:

- Mapping Administration Overview, 171
- Viewing Properties for a Mapping Job, 172
- Viewing Summary Statistics for a Mapping Job, 172
- Viewing Detailed Statistics for a Mapping Job, 172
- Viewing Logs for a Mapping Job, 173
- Reissuing a Deployed Mapping Job, 173
- Canceling a Mapping Job, 174
- Reject Files, 174

Mapping Administration Overview

When you run an ad hoc mapping job or deploy a mapping to a Data Integration Service, you can monitor the job in the Monitoring tool or the Administrator tool. If the Data Integration Service cannot write rows to the target, then you can also view information about the row in a reject file. You must have the appropriate privileges to monitor jobs or view reject files.

You can monitor a mapping job in the following locations:

- Monitoring tool. In the Developer tool, click the Menu button in the Progress view and select Monitor Jobs. Select the Data Integration Service that runs the mapping job and click OK. The Monitoring tool opens.
- Administrator tool. To monitor mappings in the Administrator tool, click the Monitor tab.

When you monitor a mapping job, you can view summary statistics or execution statistics for the job. The Summary Statistics view displays a graphical overview of the status of mapping jobs in the domain. Use the Execution Statistics view to view mapping properties and statistics, view job logs, cancel a job, or reissue a deployed mapping.
Viewing Properties for a Mapping Job

When you monitor ad hoc or deployed mapping jobs, you can view properties for the job. Properties include the job ID, the user who started the job, and the duration of the job.

1. Click the Execution Statistics view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Select the Ad Hoc Jobs folder, or expand an application and select Deployed Mapping Jobs.
   A list of jobs appears in the contents panel. The contents panel displays properties such as the name, state, ID, and duration of the jobs.
4. In the contents panel, select a job.
   The details panel displays the Properties for the job.

Viewing Summary Statistics for a Mapping Job

You can view throughput and resource usage statistics for ad hoc or deployed mapping jobs.

1. Click the Execution Statistics view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Select the Ad Hoc Jobs folder, or expand an application and select Deployed Mapping Jobs.
   A list of jobs appears in the contents panel.
4. In the contents panel, select a job.
   The details panel displays the Properties for the job.
5. Click the Summary Statistics view in the details panel.
   The Summary Statistics view displays throughput and resource usage statistics for the source and target.
   Optionally, you can sort the statistics in ascending or descending order. Click a column header to sort the column in ascending order. Click the column header again to sort the column in descending order.

Viewing Detailed Statistics for a Mapping Job

You can view graphs of the throughput and resource usage for ad hoc or deployed mapping jobs that run in separate local processes. Detailed statistics appear for jobs that run longer than one minute.

1. Click the Execution Statistics view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Select the Ad Hoc Jobs folder, or expand an application and select Deployed Mapping Jobs.
   A list of jobs appears in the contents panel.
4. In the contents panel, select a job.
   The details panel displays the Properties for the job.
5. Click the **Detailed Statistics** view in the details panel.

   The **Detailed Statistics** view displays the throughput graph and resource usage graphs.

   Optionally, you can complete the following tasks in the **Detailed Statistics** view:

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enlarge a graph</td>
<td>Move the cursor over a graph, and then click the magnifying glass icon.</td>
</tr>
<tr>
<td>Enlarge a section of an enlarged graph</td>
<td>Drag the cursor to select an area to enlarge.</td>
</tr>
<tr>
<td>Switch between rows and bytes in the throughput graph</td>
<td>Click the Bytes option or the Rows option.</td>
</tr>
<tr>
<td>Choose which statistics are plotted on the throughput graph</td>
<td>In the throughput field, select the sources and targets that you want to view.</td>
</tr>
</tbody>
</table>

### Viewing Logs for a Mapping Job

You can download the logs for a job to view the job details.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Select the **Ad Hoc Jobs** folder, or expand an application and select **Deployed Mapping Jobs**.
   
   A list of jobs appears in the contents panel.
4. In the contents panel, select a job.
5. Click **Actions > View Logs for Selected Object**.
   
   A dialog box appears with the option to open or save the log file.

### Reissuing a Deployed Mapping Job

You can reissue a deployed mapping job when the mapping jobs fails. When you reissue a deployed mapping job, the Data Integration Service runs the job again.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Expand an application and select **Deployed Mapping Jobs**.
   
   The contents panel displays a list of deployed mapping jobs.
4. Select a deployed mapping job.
5. Click **Actions > Reissue Selected Object**.
Canceling a Mapping Job

You can cancel a running ad hoc or deployed mapping job. You might want to cancel a job that stops responding or that is taking an excessive amount of time to complete.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Select the **Ad Hoc Jobs** folder, or expand an application and select **Deployed Mapping Jobs**.
   A list of jobs appears in the contents panel.
4. In the contents panel, select a job.
5. Click **Actions > Cancel Selected Object**.

Reject Files

During a mapping run, the Data Integration Service creates a reject file for each target instance in the mapping. If the Data Integration Service cannot write a row to the target, then the Data Integration Service writes the rejected row to the reject file. The reject file and mapping log contain information that helps you determine the cause of the rejection.

If the reject file does not contain any rejected rows, the Data Integration Service deletes the reject file at the end of the mapping run.

Each time you run a mapping, the Data Integration Service appends rejected data to the reject file. Depending on the source of the problem, you can correct the mapping and target database to prevent rejects in subsequent mappings.

Location of Reject Files

The Data Integration Service creates reject files for each target instance in the mapping. It creates reject files in the target reject file directory.

Configure the target reject file directory in the run-time properties for a flat file or relational target in a mapping. By default, the Data Integration Service creates reject files in the directory defined by the RejectDir system parameter. The Data Integration Service names reject files after the target instance name. The default name for reject files is `<file_name>.bad`.

When the Data Integration Service creates multiple partitions for a target, the Data Integration Service creates a separate reject file for each partition named `<file_name><partition_number>.bad`. For example, three partitions might write to reject files named MyOutput1.bad, MyOutput2.bad, and MyOutput3.bad.

Content of Reject Files

After you find a reject file, you can read it using a text editor that supports the reject file code page.

Reject files contain rows of data rejected by the writer or the target database. The Data Integration Service writes the entire row in the reject file. However, the problem usually centers on one column within the row. To help you determine which column caused the row to be rejected, the reject file contains indicators that give you more information about each column.

Reject files contain the following indicators:
**Row indicator**

The first column in each row of the reject file is the row indicator. The row indicator defines whether the row was marked for insert, update, delete, or reject.

**Column indicator**

Column indicators appear after every column of data. The column indicator defines whether the column contains valid, overflow, null, or truncated data.

**Row Indicators**

The first column in the reject file is the row indicator. The row indicator is a flag that defines the update strategy for the data row.

The following table describes the row indicators in a reject file:

<table>
<thead>
<tr>
<th>Row Indicator</th>
<th>Meaning</th>
<th>Rejected By</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Insert</td>
<td>Writer or target</td>
</tr>
<tr>
<td>1</td>
<td>Update</td>
<td>Writer or target</td>
</tr>
<tr>
<td>2</td>
<td>Delete</td>
<td>Writer or target</td>
</tr>
<tr>
<td>3</td>
<td>Reject. Marked for reject by an update strategy expression.</td>
<td>Writer</td>
</tr>
<tr>
<td>4</td>
<td>Rolled-back insert</td>
<td>Writer</td>
</tr>
<tr>
<td>5</td>
<td>Rolled-back update</td>
<td>Writer</td>
</tr>
<tr>
<td>6</td>
<td>Rolled-back delete</td>
<td>Writer</td>
</tr>
<tr>
<td>7</td>
<td>Committed insert</td>
<td>Writer</td>
</tr>
<tr>
<td>8</td>
<td>Committed update</td>
<td>Writer</td>
</tr>
<tr>
<td>9</td>
<td>Committed delete</td>
<td>Writer</td>
</tr>
</tbody>
</table>

The following sample reject file shows the row indicator of '0' for each row that indicates an insert update strategy for the row:

0,D,1921,D,Nelson,D,William,D,415-541-5145,D
0,D,1922,D,Page,D,Ian,D,415-541-5145,D
0,D,1923,D,Osborne,D,Lyle,D,415-541-5145,D
0,D,1928,D,De Souza,D,Leo,D,415-541-5145,D
0,D,2001123456789,D,S. MacDonald,D,Ira,D,415-541-514566,T
Column Indicators

A column indicator appears after every column of data. A column indicator defines whether the data is valid, overflow, null, or truncated.

The following table describes the column indicators in a reject file:

<table>
<thead>
<tr>
<th>Column Indicator</th>
<th>Type of data</th>
<th>Writer Treats As</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Valid data.</td>
<td>Good data. Writer passes it to the target database. The target accepts it unless a database error occurs, such as finding a duplicate key.</td>
</tr>
<tr>
<td>N</td>
<td>Null. The column contains a null value.</td>
<td>Good data. Writer passes it to the target, which rejects it if the target database does not accept null values.</td>
</tr>
<tr>
<td>T</td>
<td>Truncated. String data exceeded a specified precision for the column, so the value was truncated.</td>
<td>Bad data, if you configured the mapping target to reject overflow or truncated data.</td>
</tr>
</tbody>
</table>

Null columns appear in the reject file with commas marking their column. The following example shows a null column surrounded by good data:

0,D,5,D,,N,5,D

The column indicator "D" also appears after each row indicator. The following example shows the column indicator "D" after the row indicator "0":

0,D,2001123456789,O,S. MacDonald,D,Ira,D,415-541-514566,T

Either the writer or target database can reject a row. Consult the log to determine the cause for rejection.
Export to PowerCenter Overview

You can export objects from the Developer tool to use in PowerCenter.

You can export the following objects:

- **Mappings.** Export mappings to PowerCenter mappings or mapplets.
- **Mapplets.** Export mapplets to PowerCenter mapplets.
- **Logical data object models.** Export the logical data object models to PowerCenter mapplets.

You export objects to a PowerCenter repository or to an XML file. If you export objects to an XML file, PowerCenter users can import the file into the PowerCenter repository.

When you export objects, you specify export options such as the PowerCenter release, how to convert mappings and mapplets, and whether to export reference tables.

You can export mappings and mapplets that contain parameters. The parameters resolve to the default values when you import the mappings to the PowerCenter repository.
PowerCenter Release Compatibility

To verify that objects are compatible with a certain PowerCenter release, set the PowerCenter release compatibility level. The compatibility level applies to all mappings, mapplets, and logical data object models you can view in Developer tool.

You can configure the Developer tool to validate against a particular release of PowerCenter, or you can configure it to skip validation for release compatibility. By default, the Developer tool does not validate objects against any release of PowerCenter.

Set the compatibility level to a PowerCenter release before you export objects to PowerCenter. If you set the compatibility level, the Developer tool performs two validation checks when you validate a mapping, mapplet, or logical data object model. The Developer tool first verifies that the object is valid in Developer tool. If the object is valid, the Developer tool then verifies that the object is valid for export to the selected release of PowerCenter. You can view compatibility errors in the Validation Log view.

Setting the Compatibility Level

Set the compatibility level to validate mappings, mapplets, and logical data object models against a PowerCenter release. If you select none, the Developer tool skips release compatibility validation when you validate an object.

1. Click Edit > Compatibility Level.
2. Select the compatibility level.
   The Developer tool places a dot next to the selected compatibility level in the menu. The compatibility level applies to all mappings, mapplets, and logical data object models you can view in the Developer tool.

Mapplet Export

When you export a mapplet or you export a mapping as a mapplet, the export process creates objects in the mapplet. The export process also renames some mapplet objects.

The export process might create the following mapplet objects in the export XML file:

**Expression transformations**

The export process creates an Expression transformation immediately downstream from each Input transformation and immediately upstream from each Output transformation in a mapplet. The export process names the Expression transformations as follows:

Expr_{InputOrOutputTransformationName}

The Expression transformations contain pass-through ports.

**Output transformations**

If you export a mapplet and convert targets to Output transformations, the export process creates an Output transformation for each target. The export process names the Output transformations as follows:

{MappletInstanceName}_{TargetName}

The export process renames the following mapplet objects in the export XML file:
Mapplet Input and Output transformations

The export process names mapplet Input and Output transformations as follows:

<TransformationName>_<InputOrOutputGroupName>

Mapplet ports

The export process renames mapplet ports as follows:

<PortName>_<GroupName>

Mappings with Parameters Export

You can export a mapping or mapplet that contains parameters and you can import it to PowerCenter.

When you export a mapping or mapplet that contains parameters, the parameters resolve to their default values when you import the parameters to PowerCenter. The import can resolve any SQL expression that contains an parameter.

System parameters resolve to the equivalent PowerCenter system parameters. If PowerCenter does not have the equivalent system parameter, the system parameter reference remains in the mapping after you import it to PowerCenter. You need to edit the mapping and change the reference.

You cannot export mapping outputs to PowerCenter. If a mapping contains mapping outputs, the mapping is not valid in PowerCenter after you import it.

Export to PowerCenter Options

When you export an object for use in PowerCenter, you must specify the export options.

The following table describes the export options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Project in the Model repository from which to export objects.</td>
</tr>
<tr>
<td>Target release</td>
<td>PowerCenter release version.</td>
</tr>
<tr>
<td>Export selected objects to file</td>
<td>Exports objects to a PowerCenter XML file. If you select this option, specify the name and location of the export XML file.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Export selected objects to PowerCenter repository | Exports objects to a PowerCenter repository. If you select this option, specify the following connection details for the PowerCenter repository:  
- Host name. PowerCenter domain gateway host name.  
- Port number. HTTP port number of the PowerCenter domain gateway.  
- Authentication type. Select one of the following values: Kerberos Single Sign On, Native, or LDAP.  
- User name. Repository user name.  
- Password. Password for the repository user name.  
  **Note:** Specify the user name and password if the authentication type is Native or LDAP.  
- Security domain. If the authentication type is LDAP, specify the LDAP security domain name. Otherwise, enter "Native."  
- Repository name. PowerCenter repository name. |
| Send to repository folder                  | Exports objects to the specified folder in the PowerCenter repository.                                                                                                                                      |
| Use control file                           | Exports objects to the PowerCenter repository using the specified `pmrep` control file.                                                                                                                     |
| Convert exported mappings to PowerCenter mapplets | Converts Developer tool mappings to PowerCenter mapplets. The Developer tool converts data objects used as sources and targets in the mappings to Input and Output transformations in a PowerCenter mapplet. |
| Convert target mapplets                    | Converts data objects used as targets in mapplets to Output transformations in the PowerCenter mapplet. PowerCenter mapplets cannot contain targets. The export process fails if the export object includes a mapplet that contains a target and you do not select this option. |
| Export reference data                      | Exports any reference table data used by a transformation in an object you export.                                                                                                                         |
| Reference data location                    | Location for the reference table data that the Developer tool exports. The Developer tool exports the reference table data as one or more dictionary files. Enter a path to a directory on the machine that hosts the Developer tool. |
| Code page                                  | Code page of the PowerCenter repository.                                                                                                                                                                    |

### Exporting an Object to PowerCenter

When you export mappings, mapplets, or logical data object models to PowerCenter, you can export the objects to a file or to the PowerCenter repository.

Before you export an object, set the compatibility level to the appropriate PowerCenter release. Validate the object to verify that it is compatible with the PowerCenter release.

1. Click **File > Export**.  
   The **Export** dialog box appears.
2. Select **Informatica > PowerCenter**.
3. Click **Next**.  
   The **Export to PowerCenter** dialog box appears.
4. Select the project in the Model repository from which you want to export objects.

5. Select the PowerCenter release to which you want to export the objects.

6. Choose the location where you want to export the objects. You can export the objects to an XML file of the PowerCenter repository.
   - To export objects to a file, specify the name and location of an XML file.
   - To export objects to a PowerCenter repository, click Browse to specify the connection details to the repository.

7. If you export to a PowerCenter repository, select a folder in the PowerCenter repository or the pmrep control file that defines how to import objects into PowerCenter.

8. Select Convert exported mappings to PowerCenter mapplets to convert the Developer tool mappings to mapplets in PowerCenter.

9. Select Convert Target mapplets to convert data objects used as targets in a mapplet to Output transformations in the PowerCenter mapplet.

10. Select Export Reference Data to export any reference table data used by a transformation in an object you export.

11. If want to export the reference data, specify the location for the reference table data that the Developer tool exports.

12. Select the code page of the PowerCenter repository.

13. Click Next.
   The Developer tool prompts you to select the objects to export.

14. Select the objects to export and click Finish.
   The Developer tool exports the objects to the location you selected.

If you export objects to a file, you can import objects from the file into the PowerCenter repository.

If you export reference table data, copy the reference data files to the PowerCenter directory structure on the machine that hosts Informatica services. The reference data file locations must correspond to the reference table object locations in the Model repository.

For example, copy the reference data files to the following location:

<PowerCenter installation directory>\services\<Model repository project name>\<Folder name>

---

Export Restrictions

When you export a Model repository object to PowerCenter, some Model repository objects might not get exported to the PowerCenter repository. You cannot export a mapping or mapplet that contains any object that is not valid in PowerCenter.

You cannot export the following objects to PowerCenter:

**Objects with long names**

PowerCenter users cannot import a mapping, mapplet, or object within a mapping or mapplet if the object name exceeds 80 characters.

**Mappings or mapplets that contain a dynamic port**

You cannot export a mapping or mapplet that contains dynamic ports.
Mappings or mapplets that contain a Data Processor transformation

You cannot export a mapping or mapplet that contains a Data Processor transformation.

Mappings or mapplets that contain a Decision transformation that uses a system parameter

You cannot export a mapping or mapplet that contain a Decision transformation if the transformation script includes a system parameter. The export operation cannot convert the system parameter to a value that PowerCenter can use. Before you export a mapping or mapplet with a Decision transformation that uses a system parameter, replace the parameter with an appropriate value.

Mappings or mapplets that return mapping outputs

PowerCenter users cannot import a mapping or mapplet if the mapping or mapplet returns mapping output.

Mappings or mapplets that contain a Joiner transformation with certain join conditions

You cannot export mappings and mapplets that contain a Joiner transformation with a join condition that is not valid in PowerCenter. In PowerCenter, a user defines join conditions based on equality between the master and the detail sources. In the Developer tool, you can define other join conditions. For example, you can define a join condition based on the equality or the inequality between the master and the detail sources. You can define a join condition that contains transformation expressions. You can also define a join condition, such as 1 = 1, that causes the Joiner transformation to perform a cross-join.

These types of join conditions are not valid in PowerCenter. Therefore, you cannot export mappings or mapplets that contain Joiner transformations with these types of join conditions to PowerCenter.

Mappings or mapplets that contain a Lookup transformation with renamed ports

The PowerCenter Integration Service queries the lookup source based on the lookup ports in the transformation and a lookup condition. Therefore, the port names in the Lookup transformation must match the column names in the lookup source.

Mappings or mapplets that contain a Lookup transformation with certain custom SQL queries

The Developer tool uses different rules than PowerCenter to validate SQL query syntax in a Lookup transformation. A custom SQL query written in the Developer tool that uses the AS keyword or calculated fields is not valid in PowerCenter. You cannot export mappings or mapplets to PowerCenter that contain a Lookup transformation with an SQL query that uses the AS keyword or calculated fields.

Mappings or mapplets that contain sources that are not available in PowerCenter

If you export a mapping or mapplet that includes sources that are not available in PowerCenter, the mapping or mapplet fails to export.

You cannot export a mapping or mapplet with the following sources:

- Complex file data object
- DataSift
- Web Content - Kapow Katalyst

Mapplets that concatenate ports

The export process fails if you export a mapplet that contains a multigroup Input transformation and the ports in different input groups connect to the same downstream transformation.

Nested mapplets with unconnected Lookup transformations

The export process fails if you export any type of mapping or mapplet that contains another mapplet with an unconnected Lookup transformation.
Mappings with an SAP source

When you export a mapping with an SAP source, the Developer tool exports the mapping without the SAP source. When you import the mapping into the PowerCenter repository, the PowerCenter Client imports the mapping without the source. The output window displays a message indicating the mapping is not valid. You must manually create the SAP source in PowerCenter and add it to the mapping.

Mappings with Timestamp with Time Zone or Timestamp with Local Time Zone

When you import a mapping that contains data of the Timestamp with Time Zone or Timestamp with Local Time Zone data type from the Developer tool, the PowerCenter client fails to convert the mapping.

Rules and Guidelines for Exporting to PowerCenter

Due to differences between the Developer tool and PowerCenter, some Developer tool objects might not be compatible with PowerCenter.

Use the following rules and guidelines when you export objects to PowerCenter:

Verify the PowerCenter release.
Verify that the objects you want to export from the Developer tool are compatible in the target PowerCenter release.

Verify that object names are unique.
If you export an object to a PowerCenter repository, the export process replaces the PowerCenter object if it has the same name as an exported object.

Verify that the code pages are compatible.
The export process fails if the Developer tool and PowerCenter use code pages that are not compatible.

Verify precision mode.
By default, the Developer tool runs mappings and mapplets with high precision enabled and PowerCenter runs sessions with high precision disabled. If you run Developer tool mappings and PowerCenter sessions in different precision modes, they can produce different results. To avoid differences in results, run the objects in the same precision mode.

Copy reference data.
When you export mappings or mapplets with transformations that use reference tables, you must copy the reference tables to a directory where the PowerCenter Integration Service can access them. Copy the reference tables to the directory defined in the INFA_CONTENT environment variable. If INFA_CONTENT is not set, copy the reference tables to the following PowerCenter services directory:

$INFA_HOME\services\<Developer Tool Project Name>\<Developer Tool Folder Name>
Troubleshooting Exporting to PowerCenter

The export process fails when I export a mapplet that contains objects with long names.

When you export a mapplet or you export a mapping as a mapplet, the export process creates or renames some objects in the mapplet. The export process might create Expression or Output transformations in the export XML file. The export process also renames Input and Output transformations and mapplet ports.

To generate names for Expression transformations, the export process appends characters to Input and Output transformation names. If you export a mapplet and convert targets to Output transformations, the export process combines the mapplet instance name and target name to generate the Output transformation name. When the export process renames Input transformations, Output transformations, and mapplet ports, it appends group names to the object names.

If an existing object has a long name, the exported object name might exceed the 80 character object name limit in the export XML file or in the PowerCenter repository. When an object name exceeds 80 characters, the export process fails with an internal error.

If you export a mapplet, and the export process returns an internal error, check the names of the Input transformations, Output transformations, targets, and ports. If the names are long, shorten them.
Import From PowerCenter

This chapter includes the following topics:

- Import from PowerCenter Overview, 185
- Override Properties, 186
- Conflict Resolution, 186
- Import Summary, 186
- Datatype Conversion, 187
- Transformation Conversion, 187
- Mappings with Parameters Import, 194
- PowerCenter Repository Connection Parameters, 195
- Connection Assignments, 196
- Importing an Object from PowerCenter, 197
- Import Restrictions, 198
- Import Performance, 198

Import from PowerCenter Overview

You can import objects from a PowerCenter repository to a Model repository. The import process validates and converts PowerCenter repository objects to Model repository objects and imports them.

When you import objects from PowerCenter, you select the objects that you want to import and the target location in the Model repository. The import process provides options to resolve object name conflicts during the import.

You can also choose to assign connections in the Model repository to the PowerCenter objects. You can assign a single connection to multiple PowerCenter objects at the same time.

You can import mappings that contain parameters. When you import a mapping with reusable transformations, the import process imports the PowerCenter mapping parameters and generates the transformation-level parameters to bind them to. If the mapping has nonreusable transformations, the input process creates the parameters at the mapping level.

After the import process completes, you can view the import summary.
Override Properties

You can choose to preserve or ignore the override properties of PowerCenter objects during the import process. By default, the import process preserves the override properties of PowerCenter objects.

When you preserve the override properties, the import process creates nonreusable transformations or reusable data objects for the PowerCenter objects. If a PowerCenter mapping overrides source and target properties, the import process creates a data object with the same override property values as the PowerCenter mapping. The import process appends a number to the name of the PowerCenter object and creates the data object.

Conflict Resolution

You can resolve object name conflicts when you import an object from PowerCenter and an object with the same name exists in the Model repository.

You can choose from the following conflict resolution options:

- **Rename object in target**: Renames the PowerCenter repository object with the default naming convention, and then imports it. The default conflict resolution is to rename the object.
- **Replace object in target**: Replaces the Model repository object with the PowerCenter repository object.
- **Reuse object in target**: Reuses the object in the Model repository in the mapping.

**Important**: The Model repository does not distinguish between mappings and mapplets for conflict resolution. For example, if you import a mapplet and the repository contains mapping with the same name, you are prompted to resolve the conflict. If you choose to replace the object, the import process replaces mapping with the mapplet.

Import Summary

The import process creates an import summary after you import the PowerCenter objects into the Model repository.

You can save the import summary to a file if there are conversion errors. The import summary includes a status of the import, a count of objects that did not convert, a count of objects that are not valid after the import, and the conversion errors. You can also validate the objects after the import in the Developer tool to view the validation errors.
Datatype Conversion

Some PowerCenter datatypes are not valid in the Model repository. When you import PowerCenter objects with datatypes that are not valid, the import process converts them to valid, comparable datatypes in the Model repository.

The following table lists the PowerCenter repository datatypes that convert to corresponding Model repository datatype in the import process:

<table>
<thead>
<tr>
<th>PowerCenter Repository Datatype</th>
<th>Model Repository Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Double</td>
</tr>
<tr>
<td>Small Int</td>
<td>Integer</td>
</tr>
<tr>
<td>Nstring</td>
<td>String</td>
</tr>
<tr>
<td>Ntext</td>
<td>Text</td>
</tr>
</tbody>
</table>

Transformation Conversion

The import process converts PowerCenter transformations based on compatibility. Some transformations are not compatible with the Model repository. Others import with restrictions.

The following table describes the PowerCenter transformations that import with restrictions or that fail to import:

<table>
<thead>
<tr>
<th>PowerCenter Transformation</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Data Masking</td>
<td>Fails to import.</td>
</tr>
<tr>
<td>External Procedure</td>
<td>Fails to import.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Fails to import.</td>
</tr>
<tr>
<td>Identity Resolution</td>
<td>Fails to import.</td>
</tr>
<tr>
<td>Java</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Joiner</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Lookup</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Normalizer</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Rank</td>
<td>Imports with restrictions.</td>
</tr>
<tr>
<td>Sequence Generator</td>
<td>Imports with restrictions.</td>
</tr>
</tbody>
</table>
Some PowerCenter transformations import with restrictions based on transformation properties.
The import process might take one of the following actions based on compatibility of certain transformation properties:

- **Ignore.** Ignores the transformation property and imports the object.
- **Convert internally.** Imports the object with the transformation property but the Developer tool does not display the property.
- **Fail import.** Fails the object import and the mapping is not valid.

### Aggregator Transformation
The following table describes the import action for Aggregator transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformation Scope</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

### Java Transformation
In a Java transformation, the ports must be input ports or output ports. The import fails if the Java transformation has both input ports and output ports.
The following table describes the import action for Java transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Name</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Function Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Generate Transaction</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Inputs Must Block</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Is Partitionable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Language</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Module Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output Is Deterministic</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output Is Repeatable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Requires Single Thread Per Partition</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Runtime Location</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Update Strategy Transformation</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

Joiner Transformation

The following table describes the import action for Joiner transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null Ordering in Master</td>
<td>Convert internally.</td>
</tr>
<tr>
<td>Null Ordering in Detail</td>
<td>Convert internally.</td>
</tr>
<tr>
<td>Transformation Scope</td>
<td>Convert internally.</td>
</tr>
</tbody>
</table>

Lookup Transformation

The following table describes the import action for Lookup transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache File Name Prefix</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Lookup Cache Initialize</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Lookup Cache Directory Name</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Transformation Property</td>
<td>Import Action</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lookup Caching Enabled</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Lookup Data Cache Size</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Lookup Index Cache Size</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Lookup Source is Static</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Lookup Sql Override</td>
<td>Ignore if converted as a stand-alone transformation, and imports to Custom SQL Query when converted within a mapping.</td>
</tr>
<tr>
<td>Lookup Source Filter</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Pre-build Lookup Cache</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Re-cache from Lookup Source</td>
<td>Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.</td>
</tr>
<tr>
<td>Recache if Stale</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Subsecond Precision</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Synchronize Dynamic Cache</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

**Normalizer Transformation**

When you import a Normalizer transformation into the Developer tool, the Normalizer transformation imports with one input group and at least one output group.

If you import a Normalizer transformation that is not part of a mapping, the Developer tool places all the input ports in the input group of the Normalizer transformation. The Developer tool creates a default output group based on the Normalizer transformation rules for the output ports. If there are no output ports in the Normalizer transformation to import, the Developer tool creates a default output group in the imported Normalizer transformation.

When the Normalizer transformation is part of a mapping, the Developer tool might create multiple output groups based on links to the downstream transformation or targets in the mapping. For more information about the rules and guidelines about links from the multi-group transformation to the target, see the *Developer Transformation Guide*.

When you import a mapping that contains a reusable Normalizer transformation, the Developer tool imports the transformation as reusable. The Developer tool also replaces the reusable Normalizer transformation instances in the mapping with non-reusable transformation instances. The Developer tool generates new links from the non-reusable Normalizer transformation to the downstream transformations and target.

In PowerCenter, the Normalizer transformation has at least one generated key port. In the Developer tool, the Normalizer transformation does not contain a generated key port. When you import a Normalizer transformation from PowerCenter, the Developer tool ignores the generated key port.
The following table describes the import action for Normalizer transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Restart</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

Sequence Generator Transformation

In PowerCenter, the Sequence Generator transformation has two ports, CURRVAL and NEXTVAL. In the Developer tool, the Sequence Generator transformation has only one port, NEXTVAL. When you import a Sequence Generator transformation in a mapping, the Developer tool creates an Expression transformation to set the values for the CURRVAL port and pass them to the downstream transformation.

For example, you have a mapping in PowerCenter with a Sequence Generator, an Aggregator, and an Expression transformation. The source data passes to the Expression transformation followed by the Aggregator transformation and to the target. The Sequence Generator transformation adds a sequence number to each row with the CURRVAL port.

The following image shows the PowerCenter mapping with a Sequence Generator transformation, Aggregator transformation, and an Expression transformation:

The following image shows the mapping imported into the Developer tool:
When you import the mapping, the Developer tool passes the NEXTVAL data from the Sequence Generator transformation to an Expression transformation to set the CURRVAL value.

The Developer tool imports a reusable Sequence Generator transformation in the mapping as a non-reusable transformation with a reusable sequence data object. When you import a reusable Sequence Generator transformation that is not part of the mapping, the Developer tool creates a reusable sequence data object.

The following table describes the import action for Sequence Generator transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Value</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Number of Cached Values</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

**Sorter Transformation**

The following table describes the import action for Sorter transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformation Scope</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

**Source Qualifier Transformation**

The following table describes the import action for Source Qualifier transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Sorted Ports</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>
## SQL Transformation

The following table describes the import action for SQL transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Commit</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Connection Type</td>
<td>Fail import if set to dynamic connection object or full dynamic connect information.</td>
</tr>
<tr>
<td>Database Type</td>
<td>Fail import for Sybase, Informix, or Teradata.</td>
</tr>
<tr>
<td>Function Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Generate Transaction</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Inputs must Block</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Is Partitionable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Language</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Maximum Connection Pool</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Module Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output is Deterministic</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output is Repeatable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Requires single Thread Per Partition</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Runtime Location</td>
<td>Ignore.</td>
</tr>
<tr>
<td>SQL Mode</td>
<td>Fail import for script mode.</td>
</tr>
<tr>
<td>Transformation Scope</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Treat DB Connection Failure as Fatal</td>
<td>Convert internally.</td>
</tr>
<tr>
<td>Update Strategy</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Transformation</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Use Connection Pool</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>
Union Transformation

The following table describes the import action for Union transformation properties:

<table>
<thead>
<tr>
<th>Transformation Property</th>
<th>Import Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Name</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Function Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Generate Transaction</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Inputs Must Block</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Is Partitionable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Language</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Module Identifier</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output Is Deterministic</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Output Is Repeatable</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Requires Single Thread Per Partition</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Runtime Location</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Transformation Scope</td>
<td>Ignore.</td>
</tr>
<tr>
<td>Update Strategy Transformation</td>
<td>Ignore.</td>
</tr>
</tbody>
</table>

Mappings with Parameters Import

You can import a PowerCenter mapping or mapplet that contains parameters.

When a PowerCenter mapping or mapplet contains parameters, the import process creates the parameter bindings between the PowerCenter mapping parameters and the reusable transformations that reference the parameters. The import process generates the parameters at the transformation level.

PowerCenter does not require initial values for parameters. When you import a PowerCenter parameter with no initial values, the import process assigns a default value for the parameter based on the parameter data type.

If you import a mapping with variables, the import process does not convert the variables. The mapping conversion might be successful, but the mapping is not valid in the Model repository. You can change the mapping to use parameters instead of variables in the Model repository.

Some PowerCenter mapping parameters have the `IsExprVar` property enabled. This property indicates that the PowerCenter Integration Service must expand the parameter before parsing an expression. The `IsExprVar` property is not valid in the Model repository. If you import a mapping with this parameter property enabled, the conversion is successful, but the mapping is not valid.
Mappings with System-Defined Parameters Import

You can import a PowerCenter mapping or mapplet that contains some system-defined parameters. The import process imports the parameters if they match the system-defined parameters that are valid in the Model repository.

The import process creates the parameter bindings between the system-defined parameters and the reusable transformations that reference the parameters.

If a PowerCenter mapping has a system-defined parameter that does not have an equivalent system-defined parameter in the Model repository, the conversion does not fail. The import process copies the mapping property with the parameter name as the property value. However, the imported mapping is not valid. You can create a user-defined parameter to replace the property value or you can change the mapping logic.

You can import the following system-defined parameters:

- `$PMMappingName`
- `$PMIntegrationServiceName`
- `$PMRepositoryUserName`
- `$SESSSTARTTIME`
- `$SYSDATE`

You cannot import the following PowerCenter system-defined parameters:

- `$PMFolderName`
- `$PMRepositoryServiceName`
- `$PMSessionName`
- `$PMSessionRunMode`
- `$PMTAB_ALL_DATA_TYPES@TableName`
- `$PMTGT_ALL_DATA_TYPES@TableName`
- `$PMWorkflowName`
- `$PMWorkflowRunId`
- `$PMWorkflowRunInstanceName`

PowerCenter Repository Connection Parameters

When you import objects from a PowerCenter repository, you must specify the connection parameters to the repository. The Developer tool uses the import parameters to connect to the PowerCenter repository.

The following table describes the import parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>PowerCenter domain gateway host name.</td>
</tr>
<tr>
<td>Port Number</td>
<td>PowerCenter domain gateway HTTP port number.</td>
</tr>
<tr>
<td>Release Number</td>
<td>PowerCenter release version.</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication Type</td>
<td>The type of user authentication required to connect to the PowerCenter repository. Select one of the following values: Kerberos Single Sign On, Native, or LDAP. <strong>Note:</strong> Specify the user name and password if the authentication type is Native or LDAP.</td>
</tr>
<tr>
<td>User Name</td>
<td>PowerCenter repository user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Password for the PowerCenter repository user name.</td>
</tr>
<tr>
<td>Security Domain</td>
<td>If authentication type is LDAP, specify the LDAP security domain name. If not, enter Native.</td>
</tr>
<tr>
<td>Repository Name</td>
<td>PowerCenter repository name.</td>
</tr>
<tr>
<td>Code Page</td>
<td>Code page of the PowerCenter repository.</td>
</tr>
</tbody>
</table>

### Connection Assignments

When you import data sources and other objects from PowerCenter, you can assign a data source connection type.

For example, you can create a source definition, a target definition, or a Lookup transformation in PowerCenter that connects to an Oracle database. When you import these objects in the Developer tool, use the **Connection Assignment** dialog box to specify the connection type for each object.

When you select a connection type for PowerCenter repository objects, use one of the following methods to assign connections:

- **Assign a single connection to multiple PowerCenter objects at the same time.**

  You can assign a single connection to all sources, all targets, all Lookup transformations, or all objects that do not have an assigned connection. Or, you can assign a single connection to all objects with names that match a specified name pattern. Select an option from the **Select** list and click **Assign Connection**.

- **Assign a single connection to multiple PowerCenter objects of different object types.**

  Choose the **Custom** option in the **Select** list, select multiple PowerCenter objects, and then click **Assign Connection**.

- **Assign a connection to a PowerCenter object.**

  Select a PowerCenter object and click the **Open** button in the **Connection Name** column.

You can assign a connection to an object that is different from the original connection type. You might do this if Developer tool does not support the original connection type. If you assign a different connection type than the original one, the Developer tool warns you that the connection types do not match.

If you choose to ignore the warning and proceed, the import succeeds and assigns the new connection to imported data sources. The source or target are valid when the metadata for the chosen connection matches the schema of the imported data sources.
Note: For Lookup transformations created with a connection type that the Developer tool does not support, information about the original connection type is not available, and the Developer tool does not warn about a connection mismatch.

Importing an Object from PowerCenter

You can import objects from a PowerCenter repository to a Model repository.

Connect to the target Model repository before you import objects from PowerCenter.

1. Select File > Import.
The Import dialog box appears.
2. Select Informatica > PowerCenter.
3. Click Next.
   The Import from PowerCenter dialog box appears.
4. Specify the connection parameters for the PowerCenter repository.
5. Click Test Connection.
The Developer tool tests the connection to the PowerCenter repository.
6. If the connection to PowerCenter repository is successful click OK. Click Next.
The Developer tool displays the folders in the PowerCenter repository and prompts you to select the objects to import.
7. Select one or more objects to import.
8. Click Next
9. Select a target location for the import objects in the Model repository.
10. Select a conflict resolution option for object name conflict. Choose to rename, replace, or reuse the object in the target Model repository.
    • To rename the PowerCenter repository object with the default naming convention, and then import it to the Model repository, select the Rename object in target option. The default conflict resolution is to rename the object.
    • To replace the Model repository object with the PowerCenter repository object, select the Replace object in target option.
    • To reuse the object in the Model repository in the mapping instead of importing the PowerCenter object, select the Reuse object in target option.
11. Click Next.
The Developer tool shows the PowerCenter objects, and the dependent objects.
12. Click Ignore Overridden Properties to ignore override properties for reusable PowerCenter sources, targets, and transformations. By default, the process preserves override properties.
13. If you import an IBM DB2 object, select the DB2 object type. You can select one of the following object type: LOW, z/OS, i5/OS.
14. Click Next.
15. Specify the Model repository connection details to the PowerCenter repository objects.
16. The Choose Connection dialog box appears. Select a connection and click OK.
17. Click Next.
   The Developer tool generates an import summary and lists the PowerCenter objects and the dependent
   objects to be imported.

18. Click Conversion Check to verify if objects can be imported as valid Model repository objects.
   The Developer tool shows a conversion check summary report with results of the conversion check.

19. Click OK. Click Finish.
   The Developer tool shows progress information during the import. The Developer tool imports the
   PowerCenter objects and the dependent objects to the Model repository, and generates a final import
   summary report.

20. Click Save and specify a file name to save the import summary if there are conversion errors.

### Import Restrictions

The following restrictions apply when you import PowerCenter objects:

**Source and Target**
- When you import a source or target from PowerCenter release 9.1.0 or earlier, the import process
cannot verify if a connection type associated with the object is valid.
- If the version of the PowerCenter repository is earlier than 9.5.0, an IBM DB2 source database name
  or an IBM DB2 target name must start with "DB2" to set the DB2 type.
- When the row delimiter for a flat file source is not valid, the import process changes it to the default
  value.

**Transformation**
- An expression in a transformation must contain 4,000 characters or fewer.
- The database type for an SQL transformation or a Lookup transformation converts to ODBC during
  the import process.
- When you set the data cache size or the index cache size for a transformation to a value that is not
  valid, the import process changes the value to Auto.

**Mapping**
- A mapping must contain only one pipeline.

### Import Performance

If you want to import mappings larger than 68 MB, import the mapping through command line for optimal
performance.

**Tip:** You can use the following command line option: `ImportFromPC`
CHAPTER 10

Performance Tuning

This chapter includes the following topics:

- Performance Tuning Overview, 199
- Optimization Methods, 200
- Optimizer Levels, 204
- Setting the Optimizer Level for a Developer Tool Mapping, 205
- Setting the Optimizer Level for a Deployed Mapping, 206

Performance Tuning Overview

The Data Integration Service optimizes mappings to improve the performance of a mapping.

The Data Integration Service can perform the following optimizations:

**Filter data to reduce the number of rows to be processed.**

The Data Integration Service applies optimization methods in an attempt to reduce the amount of data to process. When you run a mapping, you can choose an optimizer level that determines which optimization methods the Data Integration Service can apply to the mapping. For example, the Data Integration Service can use early selection optimization to move a filter closer to the source. It can use pushdown optimization to push transformation logic to a database. It can use the cost-based optimization method to change the join processing order.

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, early selection, branch pruning, or push-into optimization methods when you select the normal optimizer level.

**Determine the partitioning strategy to maximize parallel processing.**

If you have the partitioning option, the Data Integration Service can maximize parallelism for mappings. The Data Integration Service dynamically determines the partitioning strategy for mappings. The partitioning strategy includes the location of partition points, the optimal number of partitions for each pipeline stage, and the partitioning types that best redistribute data across each partition point. For more information about partitioning, see "Partitioned Mappings Overview" on page 228.

You can also set constraints on relational sources, logical data objects, physical data objects, and virtual tables in a mapping to filter unnecessary rows. The Data Integration Service can process constraints to improve mapping performance.
Optimization Methods

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 optimization
- Early selection optimization
- Branch pruning optimization
- Push-into optimization
- Predicate optimization
- Global predicate optimization
- Cost-based optimization
- Dataship-join optimization
- Semi-join optimization

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 optimization, predicate optimization, global predicate optimization, branch pruning optimization, and early selection optimization 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.

The early projection optimization method 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 transformation that has side effects.
- 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 early selection optimization method is enabled by default when you choose the normal or full optimizer level in the Developer tool. The Data Integration Service ignores early selection optimization if a transformation that appears before the Filter transformation has side effects. The Data Integration Service cannot determine if the SQL transformation, Web Service Consumer transformation, and Java transformation have side effects. You can configure early selection optimization for these transformations if they do not have side effects.

You can disable early selection if the optimization does not increase performance. The Data Integration Service enables this optimization method by default.

Branch Pruning Optimization Method

The Data Integration Service can apply the branch pruning optimization method to transformations that do not contribute any rows to the target in a mapping.

The Data Integration Service might remove a Filter transformation if the filter condition evaluates to FALSE for the data rows. For example, a mapping has two Filter transformations that filter data from two relational sources. A Filter transformation has the filter condition Country=US, and the other Filter transformation has the filter condition Country=Canada. A Union transformation joins the two relational sources and has the filter condition Country=US. The Data Integration Service might remove the Filter transformation with the filter condition Country=Canada from the mapping.

The Developer tool enables the branch pruning optimization method by default when you choose the normal or full optimizer level. You can disable branch pruning if the optimization does not increase performance by setting the optimizer level to minimal or none.

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. The Data Integration Service applies this optimization method by default.

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 possible performance. Cost-based optimization reduces run time for mappings that perform adjacent inner-join, and full-outer 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.

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 can also apply a sorted merge join if it determines that the sorted merge join performance is better than the nested loop join performance. A sorted merge join uses sort order to arrange two data sets before performing the join. A nested loop join uses nested loops to join two data sets. The Data Integration Service might use the sorting information in the sources or create a Sorter transformation if the cost of sorting the data is less expensive then processing the nested loop join.

The Data Integration Service generates all or almost all of the mappings that are semantically equivalent to the original mapping. It uses profiling statistics 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.

The Developer tool does not enable this method by default.

Dataship-Join Optimization Method

The dataship-join optimization method attempts to locate smaller data sets next to larger data sets to reduce join processing time. The Data Integration Service attempts to apply the dataship-join optimization method when there is a significant size difference between two tables.

For example, the Data Integration Service can apply the dataship-join optimization method to join a master table that contains 10,000 rows with a detail table that contains 1,000,000 rows. To perform the dataship-join, the Data Integration Service creates a temporary staging table in the database that contains the larger detail table. Then, the Data Integration Service copies the smaller master table to a temporary table and joins the data in the temporary table with the data in the larger detail table. After the Data Integration Service performs the join operation, the Joiner transformation logic is processed in the database.
Before applying the dataship-join optimization method, the Data Integration Service performs analyses to determine whether dataship-join optimization is possible and likely to be worthwhile. If the analyses determine that this method is likely to improve 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 dataship-join optimization. It performs additional optimizations if appropriate.

The Developer tool does not enable this method by default.

**Dataship-Join Requirements for Increased Performance**

The dataship-join optimization method does not always increase performance. The following factors affect mapping performance with dataship-join optimization:

- The Joiner transformation master source must have significantly fewer rows than the detail source.
- The detail source must be significantly large to justify the optimization. If the detail source is not large enough the Data Integration Service finds it is faster to read all the data from the master and detail source without applying the dataship-join optimization method.

**Dataship-Join Optimization Rules and Guidelines**

The Data Integration Service can apply dataship-join optimization to a Joiner transformation if the transformation meets the following requirements:

- The join type must be normal, master outer, or detail outer.
- The detail pipeline must originate from a relational source.
- If the mapping uses target-based commits, the Joiner transformation scope must be All Input.
- The master and detail pipelines cannot share any transformation.
- The mapping cannot contain a branch between the detail source and the Joiner transformation.
- The Data Integration Service fails to apply the dataship-join optimization method if the database which contains the detail side of the join is an IBM DB2 database that does not support Unicode encoding.

**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 the semi-join optimization 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 the semi-join 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 improve 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 Developer tool does not enable this method by default.
Semi-Join Optimization Requirements for Increased Performance

The semi-join optimization method does not always increase performance. The following factors affect mapping performance with semi-join optimization:

- The Joiner transformation master source must have significantly fewer rows than the detail source.
- The detail source must be large enough to justify the optimization. When the Data Integration Service applies semi-join optimization, the method adds some overhead time to mapping processing. If the detail source is small, the time required to apply the semi-join method might exceed the time required to process all rows in the detail source.
- The Data Integration Service must be able to obtain source row count statistics for a Joiner transformation in order to accurately compare the time requirements of the regular join operation against the semi-join operation.

Semi-Join Optimization Rules and Guidelines

The Data Integration Service can apply semi-join optimization to a Joiner transformation if the transformation meets the following requirements:

- The join type must be normal, master outer, or detail outer. The joiner transformation cannot perform a full outer join.
- The detail pipeline must originate from a relational source.
- The join condition must be a valid sort-merge-join condition. That is, each clause must be an equality of one master port and one detail port. If there are multiple clauses, they must be joined by AND.
- If the mapping does not use target-based commits, the Joiner transformation scope must be All Input.
- The master and detail pipelines cannot share any transformation.
- The mapping cannot contain a branch between the detail source and the Joiner transformation.

Viewing an Optimized Mapping

You can view an optimized mapping to determine how the optimization methods affect the mapping.

- Right-click an empty area in the editor and click **Show Optimized Mapping**.
  The Data Integration Service generates the optimized mapping.

Optimizer Levels

The Data Integration Service optimizes mappings based on the optimizer level that you configure. Configure the optimizer level when you want the mapping to use an optimizer level other than the normal. By default, each mapping uses the normal optimizer level.

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, branch pruning, push-into, global predicate optimization, and predicate optimization methods. Normal is the default optimization level.

Full

The Data Integration Service applies the cost-based, early projection, early selection, branch pruning, predicate, push-into, semi-join, and dataship-join optimization methods.

The Data Integration Service applies the normal optimizer level when you run a mapping from the Run menu or mapping editor in the Developer tool. When you run the mapping from the Run menu, the Data Integration Service applies the optimizer level in the mapping configuration. When you run the mapping from the command line, the Data Integration Service applies the optimization level from the mapping deployment properties in the application.

Note: The Data Integration Service does not apply the pushdown optimization method with an optimizer level. You can configure pushdown optimization for a mapping in the mapping run-time properties.

Related Topics:
- “Pushdown Optimization Overview” on page 207

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.
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 that 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.
CHAPTER 11

Pushdown Optimization

This chapter includes the following topics:

- **Pushdown Optimization Overview**, 207
- **Pushdown Types**, 208
- **Transformation Logic Pushdown**, 210
- **Pushdown Optimization to Sources**, 211
- **Pushdown Optimization Expressions**, 214
- **Comparing the Output of the Data Integration Service and Sources**, 226

Pushdown Optimization Overview

When the Data Integration Service applies pushdown optimization, it pushes 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 runs the SQL queries to process the transformations.

Pushdown optimization improves mapping performance 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.

When you configure pushdown optimization for the mapping, 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 for each source that has transformation logic pushed down. Then, it reads the results of this SQL query and processes the remaining transformations in the mapping.
Pushdown Types

The Data Integration Service applies pushdown optimization to a mapping when you select the pushdown type in the mapping run-time properties.

You can select the following pushdown types:

- None. Select no pushdown type for the mapping.
- Source. The Data Integration Service tries to push down as much transformation logic as it can to the source database.
- Full. The Data Integration Service pushes the full transformation logic to the source database.

You can also create a string parameter for the pushdown type and use the following parameter values:

- Full
- Source
- None

Full Pushdown Optimization

When the Data Integration Service applies full pushdown optimization, it pushes all the transformation logic in the mapping to the source database. You can configure full pushdown in the mapping run-time properties.

Full pushdown optimization is ideal when the source and target are in the same database or when transformations such as Aggregator and Filter transformations are processed in the source database and reduce the amount of data moved. For example, if a mapping contains a Teradata source and Teradata target, configure full pushdown optimization to push all the transformation logic for processing from a Teradata source database to a Teradata target database.

When you configure a mapping with an Update Strategy transformation for full pushdown, you must determine pushdown compatibility for the mapping.

The Data Integration Service can pushdown a mapping with an Update Strategy transformation in the following scenarios:

- If the target transformation connected to the Update Strategy transformation receives multiple rows that do not have the same key.
- If the target transformation connected to the Update Strategy transformation receives multiple rows with the same key that can be reordered.

The Data Integration Service cannot pushdown a mapping with an Update Strategy transformation in the following scenario:

- If the target transformation connected to the Update Strategy transformation receives multiple rows with the same key that cannot be reordered.

You can also use a pushdown compatibility parameter in the mapping. You can use the following parameter values:

- noMultipleRowsWithSameKeyOnTarget
The Data Integration Service can use full pushdown optimization for the following sources:

- Oracle
- IBM DB2
- Microsoft SQL Server
- Teradata
- Netezza
- Greenplum
- SAP HANA

Rules and Guidelines for Full Pushdown Optimization

Consider the following rules and guidelines when you configure full pushdown optimization:

- The Data Integration Service can push all transformation logic in the mapping to IBM DB2, Oracle, Microsoft SQL Server, and ODBC sources such as Teradata, Greenplum, Netezza, and SAP HANA.
- When you configure full pushdown optimization for a mapping with an Update Strategy transformation, you can use the Update else Insert strategy only for Oracle and Teradata.

Source Pushdown

When the Data Integration Service applies source pushdown, it analyzes the mapping from source to target or until it reaches a downstream transformation 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 it can push to the database. Then, it reads the results of this SQL query and processes the remaining transformations.

You can configure a mapping to use source pushdown if the source and target reside in different databases. For example, if a mapping contains a Teradata source and an Oracle target, you can configure source pushdown to push some transformation logic for processing to the Teradata source.

Configuring Pushdown

You can configure a mapping for pushdown optimization in the mapping run-time properties.

1. Open a mapping.
2. On the Properties tab, select Run-time.
3. Choose a pushdown type or assign a pushdown parameter:
   - None. The Data Integration Service does not pushdown the mapping logic to the source database.
   - Full. The Data Integration Service pushes down the full mapping logic to the source database.
   - Source. The Data Integration Service pushes down all mapping logic except the target to the source database.
   - Assign Parameter. Select the parameter that you configured for pushdown type or create a new parameter and click OK.
4. Optionally, if you choose full pushdown optimization and the mapping contains an Update Strategy transformation, you can choose a pushdown compatibility option or assign a pushdown compatibility parameter.

- **Multiple rows do not have the same key.** The target transformation connected to the Update Strategy transformation receives multiple rows that have the same key. The Data Integration Service can pushdown the Update Strategy transformation.

- **Multiple rows with the same key can be reordered.** The target transformation connected to the Update Strategy transformation receives multiple rows with the same key that can be reordered. The Data Integration Service can pushdown the Update Strategy transformation.

- **Multiple rows with the same key cannot be reordered.** The target transformation connected to the Update Strategy transformation receives multiple rows with the same key that cannot be reordered. The Data Integration Service cannot pushdown the Update Strategy transformation.

- **Assign Parameter.** Select the parameter that you configured for pushdown compatibility or create a parameter and click OK.

---

**Transformation Logic Pushdown**

The Data Integration Service uses pushdown optimization to push transformation logic to source databases. 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:

- Aggregator
- Expression
- Filter
- Joiner
- Lookup
- Sorter
- Union

The Data Integration Service cannot push transformation logic to a source in the following circumstances:

- The source contains a column with a binary data type.
- The source is a customized data object that contains a filter condition or user-defined join for Expression or Joiner transformation logic.
- The sources are on different database management systems or use different connections for Joiner or Union transformation logic.

The Data Integration Service processes mapping logic that it cannot push to the source.
Pushdown Optimization to Sources

The Data Integration Service can push transformation logic to different sources such as relational sources and sources that use database-specific ODBC drivers. The type of transformation logic that the Data Integration Service pushes depends on the source type.

The Data Integration Service can push transformation logic to the following types of sources:

- Relational sources
- Sources that use native database drivers
- PowerExchange nonrelational sources
- Sources that use database-specific ODBC drivers
- SAP sources

Pushdown Optimization to Relational Sources

The Data Integration Service can push transformation logic to relational sources using the native drivers or database-specific ODBC drivers.

The Data Integration Service can push Aggregator, Expression, Filter, Joiner, Sorter, and Union transformation logic to the following relational sources:

- Greenplum
- Hive
- IBM DB2
- Microsoft SQL Server
- Oracle
- SAP HANA
- Sybase
- Teradata

When you push Aggregator transformation logic to a relational source, pass-through ports are valid if they are group-by ports. The transformation language includes aggregate functions that you can use in an Aggregator transformation.

The following table displays the aggregate functions that are valid in an IBM DB2 relational source:

<table>
<thead>
<tr>
<th>Aggregate Functions</th>
<th>DB2-LUW</th>
<th>DB2i</th>
<th>DB2z/os</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FIRST</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LAST</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MAX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### Aggregate Functions

<table>
<thead>
<tr>
<th>Aggregate Functions</th>
<th>DB2-LUW</th>
<th>DB2i</th>
<th>DB2z/os</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIN</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PERCENTILE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STDDEV</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following table displays the aggregate functions that are valid in Greenplum, Hive, MSSQL, Oracle, Sybase, and Teradata relational sources:

<table>
<thead>
<tr>
<th>Aggregate Functions</th>
<th>Greenplum</th>
<th>Hive</th>
<th>MSSQL</th>
<th>Oracle</th>
<th>Sybase</th>
<th>Teradata</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FIRST</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LAST</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MAX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MIN</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PERCENTILE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STDDEV</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

A relational source has a default configuration for treating null values. By default, some databases treat null values lower than any other value and some databases treat null values higher than any other value. You can push the Sorter transformation logic to the relational source and get accurate results when the source has the default null ordering.

If you configure a Sorter transformation for distinct output rows, you must enable case sensitive sorting to push transformation logic to source for DB2, Sybase, and Oracle.

The Data Integration Service cannot push any function that contains the Decimal data type to a Hive source.
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 Aggregator, Expression, Filter, Joiner, Sorter, and Union 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

The Data Integration Service can push Filter transformation logic to the following native sources:

- IBM DB2 for i5/OS
- IBM DB2 for z/OS

Pushdown Optimization to PowerExchange Nonrelational Sources

For PowerExchange nonrelational data sources on z/OS systems, the Data Integration Service pushes Filter transformation logic to PowerExchange. PowerExchange translates the logic into a query that the source can process.

The Data Integration Service can push Filter transformation logic for the following types of nonrelational sources:

- IBM IMS
- Sequential data sets
- VSAM

Pushdown Optimization to ODBC Sources

The Data Integration Service can push transformation logic to databases that use database-specific ODBC drivers. If you select the ODBC provider as Other, the Data Integration Service cannot push transformation logic to the source.

When you use a database-specific ODBC drivers to connect to a source, the Data Integration Service uses the native database SQL to generate SQL statements.

You can specify the ODBC provider in the ODBC connection object.

You can configure a specific ODBC provider for the following ODBC connection types:

- Greenplum
- Microsoft SQL Server
- Netezza
- SAP HANA
- Sybase ASE
- Teradata
Pushdown Optimization to SAP Sources

The Data Integration Service can push Filter transformation logic to SAP sources for expressions that contain a column name, an operator, and a literal string. When the Data Integration Service pushes transformation logic to SAP, the Data Integration Service converts the literal string in the expressions to an SAP datatype.

The Data Integration Service can push Filter transformation logic that contains the TO_DATE function when TO_DATE converts a DATS, TIMS, or ACCP datatype character string to one of the following date formats:

- ‘MM/DD/YYYY’
- ‘YYYY/MM/DD’
- ‘YYYY-MM-DD HH24:MI:SS’
- ‘YYYY/MM/DD HH24:MI:SS’
- ‘MM/DD/YYYY HH24:MI:SS’

The Data Integration Service processes the transformation logic if you apply the TO_DATE function to a datatype other than DATS, TIMS, or ACCP or if TO_DATE converts a character string to a format that the Data Integration Services cannot push to SAP. The Data Integration Service processes transformation logic that contains other Informatica functions. The Data Integration Service processes transformation logic that contains other Informatica functions.

Filter transformation expressions can include multiple conditions separated by AND or OR. If conditions apply to multiple SAP tables, the Data Integration Service can push transformation logic to SAP when the SAP data object uses the Open SQL ABAP join syntax. Configure the Select syntax mode in the read operation of the SAP data object.

SAP Data Type Exceptions

The Data Integration Service processes the Filter transformation logic when the source cannot process the transformation logic and the transformation expression includes the following data types:

- RAW
- LRAW
- LCHR

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.
Informatica functions are not available for nonrelational sources on z/OS. The following table displays the Informatica functions available for pushdown optimization for IBM DB2 sources:

<table>
<thead>
<tr>
<th>Function</th>
<th>DB2 for i5/OS</th>
<th>DB2 for LUW</th>
<th>DB2 for z/OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ABS()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ADD_TO_DATE()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AES_DECRYPT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AES_ENCRYPT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ASCII()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEIL()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHOOSE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHR()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CHRCODE()</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COMPRESS()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CONCAT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COSH()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CRC32()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP_TZ()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CUME()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DATE_COMPARE()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DATE_DIFF()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DECODE_BASE64()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECOMPRESS()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ENCODE_BASE64()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Function</td>
<td>DB2 for i5/OS</td>
<td>DB2 for LUW</td>
<td>DB2 for z/OS</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ERROR()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EXP()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>FIRST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FLOOR()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>FV()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GET_DATE_PART()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GET_TIMESTAMP()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GET_TIMEZONE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GREATEST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IIF()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IN()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>INDEXOF()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>INITCAP()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>INSTR()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS_DATE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IS_NUMBER()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IS_SPACES()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ISNULL()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LAST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LAST_DAY()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LEAST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LENGTH()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOG()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOWER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LPAD()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LTRIM()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Function</td>
<td>DB2 for i5/OS</td>
<td>DB2 for LUW</td>
<td>DB2 for z/OS</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MAKE_DATE_TIME()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MAX()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MD5()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MEDIAN()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>METAPHONE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MIN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MOD()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MOVINGAVG()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MOVINGSUM()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NPER()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PERCENTILE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PMT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POWER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PV()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RAND()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RATE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REG_EXTRACT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REG_MATCH()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REG_REPLACE</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPLACECHR()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPLACESTR()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REVERSE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROUND(DATE)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ROUND(NUMBER)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RPAD()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RTRIM()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SET_DATE_PART()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Function</td>
<td>DB2 for i5/OS</td>
<td>DB2 for LUW</td>
<td>DB2 for z/OS</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>SIGN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SINH()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDEX()</td>
<td>No</td>
<td>Yes¹</td>
<td>No</td>
</tr>
<tr>
<td>SQRT()</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STDDEV()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSTIMESTAMP()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TAN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TANH()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_BIGINT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_CHAR(DATE)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_CHAR(NUMBER)</td>
<td>Yes</td>
<td>Yes²</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_DATE()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_DECIMAL()</td>
<td>Yes</td>
<td>Yes²</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_DECIMAL38()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TO_FLOAT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_INTEGER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_TIMESTAMP_TZ()</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TRUNC(DATE)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TRUNC(NUMBER)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UPPER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VARIANCE()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

¹ The Data Integration Service can push these functions to the source only when they are included in Filter transformation logic.

² When this function takes a decimal or float argument, the Data Integration Service can push the function only when it is included in Filter transformation logic.

³ When this function takes a string argument, the Data Integration Service can push the function only when it is included in Filter transformation logic.
The following table displays the Informatica functions available for pushdown optimization for Greenplum, Hive, Microsoft SQL Server, Netezza, Oracle, SAP, SAP HANA, Sybase ASE, and Teradata sources:

<table>
<thead>
<tr>
<th>Function</th>
<th>Greenplum</th>
<th>Hive</th>
<th>Microsoft SQL Server</th>
<th>Netezza</th>
<th>Oracle</th>
<th>SAP</th>
<th>SAP HANA</th>
<th>Sybase ASE</th>
<th>Teredata</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ABS()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ADD_TO_DATE()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AES_DECRYPT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AES_ENCRYPT()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ASCII()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>AVG()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEIL()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CHOOSE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHR()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CHRCODE()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COMPRESS()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CONCAT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COSH()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CRC32()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP_TZ()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CUME()</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DATE_COMPARE()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DATE_DIFF()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECODE()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DECODE_BASE64()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DECOMPRESS()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ENCODE_BASE64()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Function</td>
<td>Greenplum</td>
<td>Hive</td>
<td>Microsoft SQL Server</td>
<td>Netezza</td>
<td>Oracle</td>
<td>SAP HANA</td>
<td>Sybase ASE</td>
<td>Teradata</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>------</td>
<td>----------------------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>ERROR()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>EXP()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FIRST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FLOOR()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FV()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GET_DATE_PART()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GET_TIMESTAMP()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GET_TIMEZONE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GREATEST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IIF()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IN()</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>INDEXOF()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INITCAP()</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INSTR()</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IS_DATE()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IS_NUMBER()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IS_SPACES()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ISNULL()</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LAST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAST_DAY()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LEAST()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LENGTH()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LOG()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LOWER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LPAD()</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Greenplum</td>
<td>Hive</td>
<td>Microsoft SQL Server</td>
<td>Netezza</td>
<td>Oracle</td>
<td>SAP HANA</td>
<td>Sybase ASE</td>
<td>Teradata</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>------</td>
<td>----------------------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td><code>LTRIM()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>MAKE_DATE_TIME()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>MAX()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>MD5()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>MEDIAN()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>METAPHONE()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>MIN()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>MOD()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>MOVINGAVG()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>MOVINGSUM()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>NPER()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>PERCENTILE()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>PMT()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>POWER()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>PV()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>RAND()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>RATE()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REG_EXTRACT()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REG_MATCH()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REG_REPLACE</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REPLACECHR()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REPLACESTR()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>REVERSE()</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>ROUND(DATE)</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><code>ROUND(NUMBER)</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><code>RPAD()</code></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Greenplum</td>
<td>Hive</td>
<td>Microsoft SQL Server</td>
<td>Netezza</td>
<td>Oracle</td>
<td>SAP HANA</td>
<td>Sybase ASE</td>
<td>Teradata</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>------</td>
<td>-----------------------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>RTRIM()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SET_DATE_PART()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SIGN()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SIN()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SINH()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SOUNDEX()</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SQRT()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>STDDEV()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SUBSTR()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SUM()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SYSTIMESTAMP()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TAN()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TANH()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_BIGINT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_CHAR(DATE)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_CHAR(NUMBER)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_DATE()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_DECIMAL()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_DECIMAL38()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_FLOAT()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_INTEGER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TO_TIMESTAMP_TZ()</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TRUNC(DATE)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TRUNC(NUMBER)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Greenplum</td>
<td>Hive</td>
<td>Microsoft SQL Server</td>
<td>Netezza</td>
<td>Oracle</td>
<td>SAP HANA</td>
<td>Sybase ASE</td>
<td>Teradata</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>------</td>
<td>-----------------------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>UPPER()</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VARIANCE()</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

1. The Data Integration Service can push these functions to the source only when they are included in Filter transformation logic.
2. SYS_TIMESTAMP() supports only the SS argument.

**Hive Function Exceptions**

The Data Integration Service cannot push supported functions to Hive sources under certain conditions.

The Data Integration Service processes transformation logic for Hive sources when expressions contain supported functions with the following logic:

- LTRIM includes space as the second argument.
- RTRIM includes space as the second argument.

The Data Integration Service cannot push supported functions to Hive sources under certain conditions.

**IBM DB2 Function Exceptions**

The Data Integration Service cannot push supported functions to IBM DB2 for i5/OS, DB2 for LUW, and DB2 for z/OS 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.

Netezza Function Exceptions

The Data Integration Service cannot push supported functions to Netezza sources under certain conditions.

The Data Integration Service processes transformation logic for Netezza sources when expressions contain supported functions with the following logic:

- SYSTIMESTAMP includes dates in the YYYY-MM-DD HH24:MI:SS.US format.
- TO_CHAR(DATE) and TO_DATE() include dates in the YYYY-MM-DD HH24:MI:SS.US format with subsecond precision.

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.

The Data Integration Service cannot push the EXP() function to Teradata sources when you specify the ODBC provider in the connection object as **Other**. Set the ODBC provider to **Teradata** to push the EXP() function.
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.

Teradata Function Exceptions

The Data Integration Service cannot push supported functions to Teradata sources under certain conditions.

The Data Integration Service processes transformation logic for Teradata sources when expressions contain supported functions with the following logic:

- ADD_TO_DATE includes attributes other than YEAR and MONTH.
- IN includes the CaseFlag argument.
- INSTR includes more than two arguments.
- LTRIM includes more than one argument.
- ROUND includes more than one argument.
- RTRIM includes more than one argument.

Operators

The following table summarizes the availability of Informatica operators by source type. Each column displays whether 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>DB2 for i5/OS or z/OS</th>
<th>Greenplum</th>
<th>Hive</th>
<th>Microsoft SQL Server</th>
<th>Nonrelational*</th>
<th>Oracle</th>
<th>SAP*</th>
<th>SAP HANA</th>
<th>Sybase ASE</th>
<th>Teradata</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>/</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>%</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</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.

The output of the transformation logic can be different in the following cases:

### 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: \( \text{IF}(\text{col\_varchar2} = \text{'CA'}, \text{TRUE}, \text{FALSE}) \). 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.

SYSTIMESTAMP function

When you use the 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.
Partitioned Mappings Overview

If you have the partitioning option, administrators can enable the Data Integration Service to maximize parallelism when it runs mappings. When administrators maximize parallelism, the Data Integration Service dynamically divides the underlying data into partitions and processes all of the partitions concurrently.

If mappings process large data sets or contain transformations that perform complicated calculations, the mappings can take a long time to process and can cause low data throughput. When you enable partitioning for these mappings, the Data Integration Service uses additional threads to process the mapping which can optimize performance.

To enable partitioning, administrators and developers perform the following tasks:

Administrators set maximum parallelism for the Data Integration Service to a value greater than 1 in the Administrator tool.

Maximum parallelism determines the maximum number of parallel threads that process a single pipeline stage. Administrators increase the Maximum Parallelism property value based on the number of CPUs available on the nodes where mappings run.

Optionally, developers can set a maximum parallelism value for a mapping in the Developer tool.

By default, the Maximum Parallelism property for each mapping is set to Auto. Each mapping uses the maximum parallelism value defined for the Data Integration Service.
Developers can change the maximum parallelism value in the mapping run-time properties to define a maximum value for a particular mapping. When maximum parallelism is set to different integer values for the Data Integration Service and the mapping, the Data Integration Service uses the minimum value of the two.

When partitioning is disabled for a mapping, the Data Integration Service separates the mapping into pipeline stages and uses one thread to process each stage.

When partitioning is enabled for a mapping, the Data Integration Service uses multiple threads to process each mapping pipeline stage.

The Data Integration Service can create partitions for mappings that have physical data as input and output. The Data Integration Service can use multiple partitions to complete the following actions during a mapping run:

- Read from flat file, IBM DB2 for LUW, or Oracle sources.
- Run transformations.
- Write to flat file, IBM DB2 for LUW, or Oracle targets.

**One Thread for Each Pipeline Stage**

When maximum parallelism is set to 1, partitioning is disabled. The Data Integration Service separates a mapping into pipeline stages and uses one thread to process each stage.

Each mapping contains one or more pipelines. A pipeline consists of a Read transformation and all the transformations that receive data from that Read transformation. The Data Integration Service separates a mapping pipeline into pipeline stages and then performs the extract, transformation, and load for each pipeline stage in parallel.

Partition points mark the boundaries in a pipeline and divide the pipeline into stages. For every mapping pipeline, the Data Integration Service adds a partition point after the Read transformation and before the Write transformation to create multiple pipeline stages.

Each pipeline stage runs in one of the following threads:

- Reader thread that controls how the Data Integration Service extracts data from the source.
- Transformation thread that controls how the Data Integration Service processes data in the pipeline.
- Writer thread that controls how the Data Integration Service loads data to the target.

The following figure shows a mapping separated into a reader pipeline stage, a transformation pipeline stage, and a writer pipeline stage:

![Diagram](image)

Because the pipeline contains three stages, the Data Integration Service can process three sets of rows concurrently and optimize mapping performance. For example, while the reader thread processes the third
row set, the transformation thread processes the second row set, and the writer thread processes the first row set.

The following table shows how multiple threads can concurrently process three sets of rows:

<table>
<thead>
<tr>
<th>Reader Thread</th>
<th>Transformation Thread</th>
<th>Writer Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Set 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Row Set 2</td>
<td>Row Set 1</td>
<td>-</td>
</tr>
<tr>
<td>Row Set 3</td>
<td>Row Set 2</td>
<td>Row Set 1</td>
</tr>
<tr>
<td>Row Set 4</td>
<td>Row Set 3</td>
<td>Row Set 2</td>
</tr>
<tr>
<td>Row Set n</td>
<td>Row Set (n-1)</td>
<td>Row Set (n-2)</td>
</tr>
</tbody>
</table>

If the mapping pipeline contains transformations that perform complicated calculations, processing the transformation pipeline stage can take a long time. To optimize performance, the Data Integration Service adds partition points before some transformations to create an additional transformation pipeline stage.

### Multiple Threads for Each Pipeline Stage

When maximum parallelism is set to a value greater than 1, partitioning is enabled. The Data Integration Service separates a mapping into pipeline stages and uses multiple threads to process each stage. The number of threads in any pipeline stage equals the number of partitions in the stage.

When you maximize parallelism, the Data Integration Service dynamically performs the following tasks at run time:

**Divides the data into partitions.**

The Data Integration Service dynamically divides the underlying data into partitions and runs the partitions concurrently. The Data Integration Service determines the optimal number of threads for each pipeline stage. The number of threads used for a single pipeline stage cannot exceed the maximum parallelism value. The Data Integration Service can use a different number of threads for each pipeline stage.

**Redistributes data across partition points.**

The Data Integration Service dynamically determines the best way to redistribute data across a partition point based on the transformation requirements.
The following image shows an example mapping that distributes data across multiple partitions for each pipeline stage:

In the preceding image, maximum parallelism for the Data Integration Service is three. Maximum parallelism for the mapping is Auto. The Data Integration Service separates the mapping into four pipeline stages and uses a total of 12 threads to run the mapping. The Data Integration Service performs the following tasks at each of the pipeline stages:

- At the reader pipeline stage, the Data Integration Service queries the Oracle database system to discover that both source tables, source A and source B, have two database partitions. The Data Integration Service uses one reader thread for each database partition.

- At the first transformation pipeline stage, the Data Integration Service redistributes the data to group rows for the join condition across two threads.

- At the second transformation pipeline stage, the Data Integration Service determines that three threads are optimal for the Aggregator transformation. The service redistributes the data to group rows for the aggregate expression across three threads.

- At the writer pipeline stage, the Data Integration Service does not need to redistribute the rows across the target partition point. All rows in a single partition stay in that partition after crossing the target partition point.
Partitioned Flat File Sources

When a mapping that is enabled for partitioning reads from a flat file source, the Data Integration Service can use multiple threads to read the file source.

The Data Integration Service can create partitions for the following flat file source types:

- Direct file
- Indirect file
- Directory of files
- Command
- File or directory of files in Hadoop Distributed File System (HDFS)

When the Data Integration Service uses multiple threads to read a file source, it creates multiple concurrent connections to the source. By default, the Data Integration Service does not preserve row order because it does not read the rows in the file or file list sequentially. To preserve row order when multiple threads read from a single file source, configure concurrent read partitioning.

When the Data Integration Service uses multiple threads to read a direct file, it creates multiple reader threads to read the file concurrently.

When the Data Integration Service uses multiple threads to read an indirect file or a directory of files, it creates multiple reader threads to read the files in the list or directory concurrently. The Data Integration Service might use multiple threads to read a single file. Or, the Data Integration Service might use a single thread to read multiple files in the list or directory.

Concurrent Read Partitioning

To preserve row order when multiple threads read from a single file source, configure the Concurrent Read Partitioning property for a flat file data object to preserve the order.

Configure the Concurrent Read Partitioning property in the Advanced properties for the flat file data object. Find the property in the Runtime: Read section.

Select one of the following options for the Concurrent Read Partitioning property:

- **Optimize throughput**
  
  The Data Integration Service does not preserve row order when multiple partitions read from a single file source. Use this option if the order in which multiple partitions read from a file source is not important.
  
  Default option.

- **Keep relative order**
  
  Preserves the sort order of the input rows read by each partition.

  The following table shows an example sort order of a file source with 10 rows read by two partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Rows Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition #1</td>
<td>1,3,5,8,9</td>
</tr>
<tr>
<td>Partition #2</td>
<td>2,4,6,7,10</td>
</tr>
</tbody>
</table>
Keep absolute order

Preserves the sort order of all input rows read by all partitions. In a pass-through mapping with passive transformations, the order of the rows written to the target is in the same order as the input rows.

The following table shows an example sort order of a file source with 10 rows read by two partitions:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Rows Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition #1</td>
<td>1,2,3,4,5</td>
</tr>
<tr>
<td>Partition #2</td>
<td>6,7,8,9,10</td>
</tr>
</tbody>
</table>

Partitioned Relational Sources

When a mapping that is enabled for partitioning reads from an IBM DB2 for LUW or Oracle source, the Data Integration Service can use multiple threads to read the relational source. The Data Integration Service creates a separate connection to the database for each thread.

**Note:** If a mapping reads from a relational source other than DB2 for LUW or Oracle, the Data Integration Service uses one thread to read from the source. The Data Integration Service can use multiple threads for the remaining mapping pipeline stages.

The Data Integration Service queries the DB2 for LUW or Oracle database system for partition information. If the source tables support database partitioning, the Data Integration Service can use multiple threads to read the partitioned data from the corresponding nodes in the database. The Data Integration Service generates an SQL query for each reader thread.

The number of reader threads that the Data Integration Service uses depends on the following situations:

**Number of database partitions is less than or equal to the maximum parallelism value.**

The Data Integration Service uses one reader thread for each database partition. The Data Integration Service distributes one database partition to each reader thread.

For Oracle sources that use composite partitioning, the Data Integration Service uses one reader thread for each database subpartition. For example, if an Oracle source contains three partitions and two subpartitions for each partition, then the Data Integration Service uses six reader threads.

**Number of database partitions is more than the maximum parallelism value.**

The Data Integration Service uses the number of reader threads defined by the maximum parallelism value. The Data Integration Service distributes multiple database partitions to some of the reader threads. For example, a DB2 for LUW source has five database partitions, and the maximum parallelism value is three. The Data Integration Service uses three reader threads. The Data Integration Service distributes two database partitions to the first reader thread and the second reader thread. The service distributes one database partition to the third reader thread.

**No database partitions.**

The Data Integration Service uses one thread to read from the source. The Data Integration Service can use multiple threads for the remaining mapping pipeline stages.
Relational Connection Types for Partitioning

The Data Integration Service can use multiple threads to read a DB2 for LUW or Oracle relational source based on the connection type used to connect to the database.

You can use any of the following connection types to connect to a DB2 for LUW or Oracle database:

- DB2 for LUW connection or Oracle connection
- JDBC connection
- ODBC connection

To use multiple threads to read a DB2 for LUW or Oracle relational source, the relational data object must use a DB2 for LUW or Oracle connection.

If the DB2 for LUW or Oracle relational data object uses a JDBC or ODBC connection, the Data Integration Service uses one thread to read the source. The Data Integration Service can use multiple threads for the remaining mapping pipeline stages.

SQL Queries for Partitioned Relational Sources

When the Data Integration Service uses multiple threads to read a relational source, it generates an SQL query for each reader thread.

If the database source has more database partitions than the maximum parallelism value, the Data Integration Service distributes the data across the reader threads. The Data Integration Service can generate SQL queries that read from multiple database partitions. When an Oracle source contains subpartitions, the Data Integration Service can generate SQL queries that read from multiple database subpartitions.

DB2 for LUW or Oracle Source Example

The maximum parallelism value is three, and the relational source has five database partitions. When the Data Integration Service runs SQL queries against the database partitions, the first and second reader threads receive data from two database partitions. The third reader thread receives data from one database partition. In this example, the simple query in the Read transformation does not have the select distinct option enabled.

When you use a DB2 for LUW source, the Data Integration Service generates SQL statements similar to the following statements for the first reader thread:

```sql
SELECT <column list> FROM <table name>
WHERE (nodenumber(<column 1>)=0 OR nodenumber(<column 1>) = 3)
```

When you use an Oracle source, the Data Integration Service generates SQL statements similar to the following statements for the first reader thread:

```sql
SELECT <column list> FROM <table name> PARTITION <database_partition1 name> UNION ALL
SELECT <column list> FROM <table name> PARTITION <database_partition4 name> UNION ALL
```

Oracle Source with Subpartitions Example

An Oracle source has five partitions, 1–5, and two subpartitions, a and b, in each partition. The maximum parallelism value is three. The first reader thread receives data from four database subpartitions. The second and third reader threads receive data from three database subpartitions. In this example, the simple query in the Read transformation does not have the select distinct option enabled.

The Data Integration Service generates SQL statements similar to the following statements for the first reader thread:

```sql
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_a name> UNION ALL
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_b name>
```
Rules and Guidelines for Relational Source Partitions

Consider the following rules and guidelines when you enable partitioning for a mapping that reads from a relational source:

- The Data Integration Service uses one thread to read the source, but can use multiple threads for the remaining mapping pipeline stages in the following situations:
  - The mapping reads from a relational source other than DB2 for LUW or Oracle.
  - The mapping uses a JDBC or ODBC connection to read from a DB2 for LUW or Oracle source.
  - The mapping pushes transformation logic to the source database.
  - You use the simple query in the Read transformation to select the ports to sort by or to configure a user-defined join.
  - You use the advanced query in the Read transformation to create a custom SQL query.
- If you use the simple query in the Read transformation to create hints, select distinct values, or enter a source filter, the Data Integration Service can use multiple threads to read the source. The Data Integration Service adds the hints, distinct values, or source filter to the SQL query generated for each partition.

Partitioned Flat File Targets

When a mapping that is enabled for partitioning writes to a flat file target, the Data Integration Service can use multiple threads to write to the file target.

The Data Integration Service can create partitions for a flat file or a file in Hadoop Distributed File System (HDFS).

You can configure a flat file data object to have either a file or command output type. When a flat file data object has the file output type, the Data Integration Service writes the target data to a flat file. If multiple threads write to the flat file target, each thread writes the target output to a separate file. Each thread uses the following format to name the file:

<output_file_name><partition_number>.out

For example, three threads might write to files named MyOutput1.out, MyOutput2.out, and MyOutput3.out.

You can configure multiple output file directories to optimize performance, or you can configure the flat file data object to write to a single merge file.

When a flat file data object has the command output type, the Data Integration Service outputs the target data to a command or to a merge command instead of a flat file or a merge file. If multiple partitions write to the flat file target, you can configure a command to process target data for a single partition or to process merge data for all target partitions.
Optimize Output File Directories for Partitioned File Targets

By default when a flat file data object has a file output type, each thread writes the target output to a separate file. For optimal performance when multiple threads write to a file target, configure multiple output file directories.

When multiple threads write to a single directory, the mapping might encounter a bottleneck due to input/output (I/O) contention. An I/O contention can occur when threads write data to the file system at the same time.

When you configure multiple directories, the Data Integration Service determines the output directory for each thread in a round-robin fashion. For example, you configure a flat file data object to use directoryA and directoryB as target directories. If the Data Integration Service uses four threads to write to the file target, the first and third writer threads write target files to directoryA. The second and fourth writer threads write target files to directoryB.

If the Data Integration Service does not use multiple threads to write to the target, the service writes the output file to the first listed directory.

Configure the output file directories in the Advanced properties for the flat file data object. Find the Output File Directory property in the Runtime: Write section. By default, the property is configured to use the system parameter value defined for the Data Integration Service. Use the default TargetDir system parameter value if an administrator entered multiple directories separated by semicolons for the Target Directory property for the Data Integration Service.

You can enter a different value to configure multiple output file directories specific to the flat file data object. Enter multiple directories separated by semicolons for the property or for the user-defined parameter assigned to the property.

Merge Options for Partitioned File Targets

By default when a flat file data object has a file output type, each thread writes the target output to a separate file. You can merge target data for the partitions. When you merge target data, the Data Integration Service creates a single merge file for all target partitions.

Configure the merge options in the Advanced properties for the flat file data object. Find the merge properties in the Runtime: Write section.
The following image shows the merge options in the advanced properties for a flat file data object:

1. File output type
2. Merge options

Select one of the following options for the **Merge Type** property:

**No merge**

The Data Integration Service concurrently writes the target output to a separate file for each partition. Default option.

**Sequential**

The Data Integration Service creates an output file for each partition and then merges them into a single merge file. The Data Integration Service creates the individual target files using the output file name and output file directory values. The Data Integration Service sequentially adds the output data for each partition to the merge file, in the order that each writer thread completes. For example, if the writer thread for Partition2 finishes before the thread for Partition1, the Data Integration Service adds the data to the merge file in the following order: Partition2, Partition1.

**File list**

The Data Integration Service creates a target file for each partition and creates a file list that contains the paths of the individual files. The Data Integration Service creates the individual target files using the output file name and output file directory values. If you write the target files to the merge directory or a directory under the merge directory, the file list contains relative paths. Otherwise, the file list contains absolute paths. Use this file as a source file if you use the target files as source files in another mapping.

**Concurrent**

The Data Integration Service concurrently writes the data for all target partitions to the merge file. It does not create intermediate files for each partition. Because the Data Integration Service writes to the merge file concurrently for all partitions, the order of the data in the merge file might not be sequential.
If you configure the flat file data object to merge target data, you can optionally edit the default values for the **Merge File Directory** and **Merge File Name** properties.

If you configure the flat file data object to merge target data and the Data Integration Service does not create partitions for the target, the Data Integration Service ignores the merge options. The service writes to the file defined in the **Output File Name** and **Output File Directory** properties.

**Commands for Partitioned File Targets**

When a flat file data object has a command output type, you can use a command to process target data for a single partition or to process merge data for all target partitions in a mapping. The Data Integration Service sends the data to a command or to a merge command instead of a flat file or a merge file.

Configure the command that processes data for partitions in the **Advanced** properties for the flat file data object. Find the command properties in the **Runtime: Write** section.

The following image shows a flat file data object configured to use a command to process target data for a single partition:

1. Command output type
2. Command options

On UNIX, use any valid UNIX command or shell script. On Windows, use any valid DOS or batch file.

You can use a command to process the following types of target data:

**Data for a single partition**

Enter a command that each writer thread runs separately. Each thread sends the target data to the command when the mapping runs. Each thread runs the same command with a different set of data.

When you enter the command, you must consider the operating system on which the mapping runs. For example, if you enter the command `cat > /log/data/myoutput.txt`, multiple threads write to the same file which could cause an operating system error. If you enter the command `cat >> /log/data/`
multiple threads append data to the same file which is less likely to cause an operating system error.

To send the target data for a single partition to a command, select command for the **Output Type** property, and select no merge for the **Merge Type** property. Enter a command for the **Command** property.

**Merge data for all partitions**

Enter a merge command that processes the merge data for all writer threads. The Data Integration Service must use a concurrent merge type for a command to process merge data. Each thread concurrently sends target data to the merge command when the mapping runs. The merge command runs once with all of the data. The command might not maintain the order of the target data.

To send merge data for all partitions to a merge command, select command for the **Output Type** property, and select concurrent for the **Merge Type** property. Enter a command for the **Merge Command** property.

---

### Partitioned Relational Targets

When a mapping that is enabled for partitioning writes to an IBM DB2 for LUW or Oracle target, the Data Integration Service can use multiple threads to write to the relational target. The Data Integration Service creates a separate connection to the database for each thread.

**Note:** If a mapping writes to a relational target other than DB2 for LUW or Oracle, the Data Integration Service uses one thread to write to the target. The Data Integration Service can use multiple threads for the remaining mapping pipeline stages.

When the target is Oracle, the Data Integration Service uses the number of writer threads defined by the maximum parallelism value. If the Oracle relational table has partitions, the database routes the data to the correct partitions.

When the target is DB2 for LUW, the Data Integration Service queries the DB2 for LUW system for partition information. The Data Integration Service loads the partitioned data to the corresponding nodes in the target database.

The number of writer threads that the Data Integration Service uses for a DB2 for LUW target depends on the following situations:

- **Number of database partitions is less than or equal to the maximum parallelism value.**
  - The Data Integration Service uses one writer thread for each database partition. Each writer thread writes to one database partition.

- **Number of database partitions is more than the maximum parallelism value.**
  - The Data Integration Service uses the number of writer threads defined by the maximum parallelism value. The Data Integration Service distributes multiple database partitions to some of the writer threads. For example, a DB2 for LUW target has five database partitions, and the maximum parallelism value is three. The Data Integration Service uses three writer threads. The Data Integration Service distributes two database partitions to the first writer thread and the second writer thread. The service distributes one database partition to the third writer thread.

- **No database partitions.**
  - The Data Integration Service uses the number of writer threads defined by the maximum parallelism value.
However, you can optimize load performance when the target has database partitions. In this case, each writer thread connects to the DB2 for LUW node that contains the database partition. Because the writer threads connect to different DB2 for LUW nodes instead of all threads connecting to the single master node, performance increases.

**Relational Connection Types for Partitioning**

The Data Integration Service can use multiple threads to write to a DB2 for LUW or an Oracle relational target based on the connection type used to connect to the database.

You can use any of the following connection types to connect to a DB2 for LUW or Oracle database:

- DB2 for LUW connection or Oracle connection
- JDBC connection
- ODBC connection

To use multiple threads to write to a DB2 for LUW or Oracle relational target, the relational data object must use a DB2 for LUW or Oracle connection.

If the DB2 for LUW or Oracle relational data object uses a JDBC or ODBC connection, the Data Integration Service uses one thread to write to the target. The Data Integration Service can use multiple threads for the remaining mapping pipeline stages.

**Rules and Guidelines for Relational Target Partitions**

Consider the following rules and guidelines when you enable partitioning for a mapping that writes to a relational target:

- The Data Integration Service uses one thread to write to the target, but can use multiple threads for the remaining mapping pipeline stages in the following situations:
  - The mapping writes to a relational target other than DB2 for LUW or Oracle.
  - The mapping uses a JDBC or ODBC connection to write to a DB2 for LUW or Oracle target.
  - Enable high precision for the mapping when a DB2 for LUW target table partition key is a decimal column. The Data Integration Service might fail the mapping when a partition key is a decimal column and you do not enable high precision for the mapping.

**Partitioned Transformations**

When a mapping enabled for partitioning contains a transformation that supports partitioning, the Data Integration Service can use multiple threads to run the transformation.

The Data Integration Service determines whether it needs to add an additional partition point at the transformation, and then determines the optimal number of threads for that transformation pipeline stage. The Data Integration Service also determines whether it needs to redistribute data at the partition point. For example, the Data Integration Service might redistribute data at an Aggregator transformation to group rows for an aggregate expression.

Some transformations do not support partitioning. When a mapping enabled for partitioning contains a transformation that does not support partitioning, the Data Integration Service uses one thread to run the transformation. The Data Integration Service can use multiple threads to run the remaining mapping pipeline stages.
The following transformations do not support partitioning:

- Association
- Consolidation
- Exception
- Match, when configured for field match analysis
- REST Web Service Consumer
- Unconnected Lookup
- Web Service Consumer

**Restrictions for Partitioned Transformations**

Some transformations that support partitioning require specific configurations. If a mapping enabled for partitioning contains a transformation with an unsupported configuration, the Data Integration Service uses one thread to run the transformation. The Data Integration Service can use multiple threads to process the remaining mapping pipeline stages.

The following transformations require specific configurations to support partitioning:

- Aggregator transformations must include a group by port. Aggregator transformations must not include numeric functions that calculate running totals and averages on a row-by-row basis.
- Expression transformations must not include the following types of functions or variables:
  - Numeric functions that calculate running totals and averages on a row-by-row basis.
  - Special functions that might return different results when multiple threads process the transformation.
  - Local variables that depend on the value of a previous row.
- Decision, Java, and SQL transformations must have the **Partitionable** property enabled.
- Joiner transformations must include a join condition that uses an equality operator. If the join condition includes multiple conditions, the conditions must be combined using the AND operator.
- Rank transformations must include a group by port.

**Cache Partitioning for Transformations**

Cache partitioning creates a separate cache for each partition that processes an Aggregator, Joiner, Rank, Lookup, or Sorter transformation. During cache partitioning, each partition stores different data in a separate cache. Each cache contains the rows needed by that partition.

Cache partitioning optimizes mapping performance because each thread queries a separate cache in parallel. When the Data Integration Service creates partitions for a mapping, the Data Integration Service always uses cache partitioning for partitioned Aggregator, Joiner, Rank, and Sorter transformations. The Data Integration Service might use cache partitioning for partitioned Lookup transformations.

The Data Integration Service uses cache partitioning for connected Lookup transformations under the following conditions:

- The lookup condition contains only equality operators.
- When the connected Lookup transformation looks up data in a relational table, the database is configured for case-sensitive comparison.

For example, if the lookup condition contains a string port and the database is not configured for case-sensitive comparison, the Data Integration Service does not use cache partitioning.
When the Data Integration Service does not use cache partitioning for a Lookup transformation, all threads that run the Lookup transformation share the same cache. Each thread queries the same cache serially.

**Note:** The Data Integration Service does not use cache partitioning for unconnected Lookup transformations because the service uses one thread to run unconnected Lookup transformations.

### Cache Size for Partitioned Caches

When the Data Integration Service uses cache partitioning for Aggregator, Joiner, Rank, Lookup, and Sorter transformations, the service divides the cache size across the partitions.

You configure the cache size in the transformation advanced properties. You can enter a numeric value in bytes, or you can select **Auto** to have the Data Integration Service determine the cache size at run time.

If you enter a numeric value, the Data Integration Service divides the cache size across the number of transformation threads at run time. For example, you configure the transformation cache size to be 2,000,000 bytes. The Data Integration Service uses four threads to run the transformation. The service divides the cache size value so that each thread uses a maximum of 500,000 bytes for the cache size.

If you select **Auto**, the Data Integration Service determines the cache size for the transformation at run time. The service then divides the cache size across the number of transformation threads.

### Optimize Cache Directories for Partitioning

For optimal performance during cache partitioning for Aggregator, Joiner, Rank, and Sorter transformations, configure multiple cache directories.

Transformation threads write to the cache directory when the Data Integration Service uses cache partitioning and must store overflow values in cache files. When multiple threads write to a single directory, the mapping might encounter a bottleneck due to input/output (I/O) contention. An I/O contention can occur when threads write data to the file system at the same time.

When you configure multiple cache directories, the Data Integration Service determines the cache directory for each transformation thread in a round-robin fashion. For example, you configure an Aggregator transformation to use `directoryA` and `directoryB` as cache directories. If the Data Integration Service uses four threads to run the Aggregator transformation, the first and third transformation threads store overflow values in cache files in `directoryA`. The second and fourth transformation threads store overflow values in cache files in `directoryB`.

If the Data Integration Service does not use cache partitioning for the Aggregator, Joiner, Rank, or Sorter transformation, the service stores overflow values in cache files in the first listed directory.

**Note:** A Lookup transformation can only use a single cache directory.

Configure the cache directories in the **Cache Directory** property for the Aggregator, Joiner, or Rank transformation advanced properties. Configure the cache directories in the **Work Directory** property for the Sorter transformation advanced properties. By default, the **Cache Directory** and **Work Directory** properties are configured to use the system parameter values defined for the Data Integration Service. Use the default CacheDir or TempDir system parameter value if an administrator entered multiple directories separated by semicolons for the **Cache Directory** or **Temporary Directories** property for the Data Integration Service.

You can enter a different value to configure multiple cache directories specific to the transformation. Enter multiple directories separated by semicolons for the property or for the user-defined parameter assigned to the property.
Disable Partitioning for a Transformation

A partitioned Decision, Java, or SQL transformation might not return the same result for each mapping run. You can disable partitioning for these transformations so that the Data Integration Service uses one thread to process the transformation. The Data Integration Service can use multiple threads to process the remaining mapping pipeline stages.

In a Java or SQL transformation, the Partitionable advanced property is selected by default. Clear the advanced property to disable partitioning for the transformation.

In a Decision transformation, the Partitionable advanced property is cleared by default. Select the advanced property to enable partitioning for the transformation.

The reason that you might want to disable partitioning for a transformation depends on the transformation type.

Decision Transformation

You might want to disable partitioning for a Decision transformation that uses a numeric function. The numeric functions CUME, MOVINGSUM, and MOVINGAVG calculate running totals and averages on a row-by-row basis. If a partitioned Decision transformation includes one of these functions, each thread processes the function separately. Each function calculates the result using a subset of the data instead of all of the data. Therefore, a partitioned transformation that uses CUME, MOVINGSUM, or MOVINGAVG functions might not return the same calculated result with each mapping run.

Java Transformation

Disable partitioning for a Java transformation when the Java code requires that the transformation be processed with one thread.

SQL Transformation

Disable partitioning for an SQL transformation when the SQL queries require that the transformation be processed with one thread. Or, you might want to disable partitioning for an SQL transformation so that only one connection is made to the database.

Maintain Order in a Partitioned Mapping

You can establish order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When the Data Integration Service adds a partition point to a mapping, it might redistribute data and lose an order established earlier in the mapping. To maintain order in a partitioned mapping, you must specify that some transformations and Write transformations maintain the row order.

You can specify that the following mapping objects maintain the row order of the input data:

- Expression transformation
- Java transformation
- Sequence Generator transformation
- SQL transformation
- Write transformation

For example, if a relational target has a database trigger that depends on the data being written in the sorted order, configure the Write transformation to maintain the row order.

When you configure Write transformations to maintain row order, the Data Integration Service uses a single thread to write to the target. If an Aggregator transformation that uses sorted input precedes the Write
transformation, the Data Integration Service uses a single thread to process both the Aggregator transformation and the target.

When you configure all other transformations to maintain row order, the Data Integration Service determines the optimal number of threads for the transformation pipeline stage while maintaining the order.

The method that you use to configure transformations to maintain row order depends on the following object types:

**Expression, Sequence Generator, or SQL transformation**
- Select the **Maintain Row Order** property in the **Advanced** properties of an Expression, Sequence Generator, or SQL transformation.

**Java transformation**
- Select the **Stateless** property in the **Advanced** properties of a Java transformation.

**Write transformation**
- Select the **Maintain Row Order** property in the **Advanced** properties of the Write transformation.

### Maintain a Stable Sort

When you maintain order in a partitioned mapping, the Data Integration Service does not perform a stable sort. The Data Integration Service maintains the order of the rows based on the sort key. However, if multiple rows have equal values for the sort key, the rows might not appear in the same relative order in the output as they appear in the input.

For example, a mapping enabled for partitioning reads from a sorted flat file source that contains the following data:

```
Order_ID, Item_ID, Item, Quantity, Price
45, 000468, ItemA, 5, 0.56
45, 123456, ItemA, 5, 3.04
41, 456789, ItemB, 2, 12.02
43, 123456, ItemA, 3, 3.04
```

The mapping includes a Sorter transformation that specifies Order_ID as the sort key with the direction as descending. When the Data Integration Service uses multiple threads to run the Sorter transformation, it might not maintain the relative order of the rows with the same value for Order_ID. For example, the service might write the rows to a merged target file in the following order:

```
Order_ID, Item_ID, Item, Quantity, Price
45, 123456, ItemA, 5, 3.04
45, 000468, ItemA, 5, 0.56
43, 123456, ItemA, 3, 3.04
41, 456789, ItemB, 2, 12.02
```

To maintain a stable sort, disable partitioning for the mapping by setting the **Maximum Parallelism** run-time property for the mapping to 1.

### Override the Maximum Parallelism for a Mapping

By default, the **Maximum Parallelism** property for each mapping is set to Auto. Each mapping uses the maximum parallelism value defined for the Data Integration Service. You can override the maximum parallelism value to define a maximum value for a particular mapping.

When maximum parallelism is set to different integer values for the Data Integration Service and the mapping, the Data Integration Service uses the minimum value of the two.
You might want to override the Maximum Parallelism property for a mapping for the following reasons:

**You run a complex mapping that results in more threads than the CPU can handle.**

The total number of parallel threads that can run for the complete mapping pipeline is the parallelism value multiplied by the number of pipeline stages. Each partition point adds an additional pipeline stage. A complex mapping with multiple Aggregator or Joiner transformations might have many pipeline stages. A large number of pipeline stages can cause the Data Integration Service to use more threads than the CPU can handle.

**Mapping performance is satisfactory with fewer parallel threads for each pipeline stage.**

When a single mapping runs with fewer parallel threads, more threads are available for the Data Integration Service to run additional jobs.

**You want to define a suggested parallelism value for a transformation.**

If you override the maximum parallelism for a mapping, you can define a suggested parallelism value for a specific transformation in the mapping. You might want to define a suggested parallelism value to optimize performance for a transformation that contains many ports or performs complicated calculations.

**You want to define an execution instances value for an Address Validator or Match transformation.**

If you override the maximum parallelism for a mapping, the Data Integration Service considers the execution instances value for an Address Validator or Match transformation in the mapping. You might want to define an execution instances value to optimize performance for the transformation.

### Suggested Parallelism for a Transformation

If you override the Maximum Parallelism run-time property for a mapping, you can define the Suggested Parallelism property for a specific transformation in the mapping run-time properties.

The Data Integration Service considers the suggested parallelism value for the number of threads for that transformation pipeline stage as long as the transformation can be partitioned. For example, if you configure the mapping to maintain row order, the Data Integration Service might need to use one thread for the transformation.

If the Maximum Parallelism run-time property for the mapping is set to Auto, you cannot define a suggested parallelism value for any transformations in the mapping. The Data Integration Service ignores any suggested parallelism values previously defined for the transformations.

You might want to define a suggested parallelism value to optimize performance for a transformation that contains many ports or performs complicated calculations.

For example, if a mapping enabled for partitioning processes a small data set, the Data Integration Service might determine that one thread is sufficient to process an Expression transformation pipeline stage. However, if the Expression transformation contains many complicated calculations, the transformation pipeline stage can still take a long time to process. You can enter a suggested parallelism value greater than 1 but less than the maximum parallelism value defined for the mapping or the Data Integration Service. The Data Integration Service uses the suggested parallelism value for the number of threads for the Expression transformation.
You can configure the following values for the **Suggested Parallelism** property for a transformation when you override the maximum parallelism for the mapping:

<table>
<thead>
<tr>
<th>Suggested Parallelism Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Data Integration Service uses one thread to run the transformation.</td>
</tr>
<tr>
<td>Auto</td>
<td>The Data Integration Service considers the maximum parallelism defined for the mapping and for the Data Integration Service. The service uses the lowest value to determine the optimal number of threads that run the transformation. Default for each transformation.</td>
</tr>
<tr>
<td>Greater than 1</td>
<td>The Data Integration Service considers the suggested parallelism defined for the transformation, the maximum parallelism defined for the mapping, and the maximum parallelism defined for the Data Integration Service. The service uses the lowest value for the number of threads that run the transformation.</td>
</tr>
</tbody>
</table>

You can define the **Suggested Parallelism** property in the mapping run-time properties for the following transformations:

- Aggregator
- Expression
- Filter
- Java
- Joiner
- Lookup
- Normalizer
- Rank
- Router
- Sequence Generator
- Sorter
- SQL
- Union
- Update Strategy

**Execution Instances for Address Validator and Match Transformations**

If you override the **Maximum Parallelism** run-time property for a mapping, the Data Integration Service considers the value of the **Execution Instances** advanced property defined for an Address Validator or Match transformation.

The Data Integration Service considers the execution instances value for the number of threads for that transformation pipeline stage as long as the transformation can be partitioned. For example, if you configure the mapping to maintain row order, the Data Integration Service might need to use one thread for the transformation.

You can increase the number of execution instances on a Match transformation when you configure the transformation for identity match analysis. You cannot increase the number of execution instances on a
Match transformation when you configure the transformation for field match analysis. In field match analysis, the Match transformation uses a single execution instance.

If the Maximum Parallelism run-time property for a mapping is set to Auto, the Data Integration Service ignores the execution instances value defined for an Address Validator or Match transformation.

You can configure the following values for the Execution Instances advanced property for an Address Validator or Match transformation when you override the maximum parallelism for the mapping:

<table>
<thead>
<tr>
<th>Execution Instances Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Data Integration Service uses one thread to run the transformation. Default for the Address Validator transformation.</td>
</tr>
<tr>
<td>Auto</td>
<td>The Data Integration Service considers the maximum parallelism defined for the mapping and for the Data Integration Service. The service uses the lowest value to determine the optimal number of threads that run the transformation. Default for the Match transformation in identity match analysis.</td>
</tr>
<tr>
<td>Greater than 1</td>
<td>The Data Integration Service considers the execution instances defined for the transformation, the maximum parallelism defined for the mapping, and the maximum parallelism defined for the Data Integration Service. The service uses the lowest value for the number of threads that run the transformation.</td>
</tr>
</tbody>
</table>

**Note:** The Data Integration Service also considers the Max Address Object Count property on the Content Management Service when it calculates the optimal number of threads for an Address Validator transformation. The Max Address Object Count property determines the maximum number of address validation instances that can run concurrently in a mapping. The Max Address Object Count value must be greater than or equal to the maximum parallelism value on the Data Integration Service.

**Overriding the Maximum Parallelism Value**

To override the maximum parallelism value, set maximum parallelism in the mapping run-time properties to an integer value greater than 1 and less than the value set for the Data Integration Service.

1. Open the mapping.
2. In the Properties view, click the Run-time tab.
4. For the Maximum Parallelism property, enter an integer value greater than 1 and less than the value set for the Data Integration Service.
   Or you can assign a user-defined parameter to the property, and then define the parameter value in a parameter set or a parameter file.
5. To define a suggested parallelism value for a specific transformation in the mapping, enter an integer value greater than 1 for the transformation in the Suggested Parallelism section.
The following image shows a mapping with an overridden maximum parallelism value and with the default suggested parallelism values for transformations:

6. Save the mapping.

7. To define an execution instances value for an Address Validator or for a Match transformation configured for identity match analysis, complete the following steps:
   a. Open the Address Validator or Match transformation.
   b. In the Advanced view, enter an integer value greater than 1 for the Execution Instances property.
   c. Save the transformation.

Troubleshooting Partitioned Mappings

The solution to the following situation might help you troubleshoot partitioned mappings.

The Maximum Parallelism property for the Data Integration Service is set to a value greater than 1, and the Maximum Parallelism property for the mapping is set to Auto. However, partitioning is disabled when the mapping runs.

To run mappings with multiple partitions, the license assigned to the Data Integration Service must include partitioning. If the license does not include partitioning, the Data Integration Service writes the following message to the mapping log and uses one thread to process each mapping pipeline stage:

WARNING: The Integration Service cannot create partitions for the mapping because the license assigned to the Integration Service does not include partitioning.
INFO: Partitioning is disabled for the mapping.
INDEX

A
Address Validator transformation
  execution instances 246
aggregate function
description 72
aggregation type
mapping outputs 68
Aggregator transformation
cache partitioning 241
multiple cache directories 242
partitioned 241
application element
parameter files 54

B
binding
  mapping outputs to workflow variables 86
  mapplet outputs to mappings 76, 87
binding parameters
  instance values 63
branch pruning optimization
description 201

C
cache directory
  multiple directories 242
optimizing 242
cache partitioning
cache size 242
description 241
cache size
  partitions 242
concurrent read partitioning
description 232
constraints
  creating target load order 28
target load order 26
cost-based optimization
description 202
CUME function
  partitioning restrictions 243

data object
  get columns at run time 96
dataship-join optimization
description 202
date parameters
  valid formats 38
Decision transformation
disable partitioning 243
partitioned 241
dependencies
  implicit 20
  link path 20
dynamic expression
creating 131
dynamic expressions
  overview 113
dynamic mappings
  components 94
  configuring Write transformations 133
  creating dynamic ports 125
  creating run-time links 136
  defining input rules 126
  developing and running 123
dynamic ports 99
generated ports 99
input rules 102
overview 91
parameterize the source name 95
parameters 120
run-time links 116
running 138
sources 95
target objects 114
troubleshooting 138
validating 138
dynamic ports
  configuring 126
  creating 125
  overview 99
dynamic sources
  descriptions 95
  get columns at run time 96
early projection optimization
description 200
early selection optimization
description 201
execution instances
Address Validator transformation 246
Match transformation 246
export
to PowerCenter 177
export to PowerCenter
exporting objects 180
options 179
overview 177
parameter conversion 179
release compatibility 178
restrictions 181
mapping outputs (continued)
persisting guidelines 72
steps to configure 78
system-defined 67
user-defined 68
using infacmds 73
mapping parameters
in virtual table mappings 50
infacmd 52
overview 35
system 36
types 37
user-defined 37
where to assign 39
where to create 38
mapping pipelines
description 229
Mapping task input
binding persisted output to 85
mappings
adding objects 16
connection validation 30
creating 16
creating constraints 28
decreasing parallelism 244
developing 16
environment 23
execution environment 23
expression validation 30
maximum parallelism 23, 230
object dependency 15
object validation 30
objects 15
optimization methods 200
optimized mapping 204
overview 14
partition points 229
partitioned 230
pipelines 229
predicate optimization method 201
processing threads 229
reject files 174
run-time properties 23
running 31
stop on errors 23
target commit interval 23
target load order constraints 26
validating 31
validation 30
validation environment 23
mappings, dynamic
troubleshooting 138
mapplet outputs
binding to a mapping 76
binding to mapping outputs 87
binding to mappings 75
defining 87
how to bind to mapping 89
mapplet parameters
element 49
example 49
mapplets
creating 34
exporting to PowerCenter 178
input 33
output 34
overview 32
rules 33
types 32
mapplets (continued)
using parameters with 47
validating 34
Match transformation
execution instances 246
maximum parallelism
description 230
mapping run-time 23
overriding 245
Microsoft SQL Server sources
pushdown optimization 213
pushdown optimization 213
minimal optimization level
description 204
MOVINGAVG function
partitioning restrictions 243
MOVINGSUM function
partitioning restrictions 243
N
nonrelational sources
pushdown optimization 213
normal optimization level
description 204
O
operators
available in sources 225
pushdown optimization 225
optimization
branch pruning optimization method 201
cost-based optimization method 202
dataship-join optimization method 202
early projection optimization method 200
early selection optimization method 201
mapping performance methods 200
semi-join optimization method 203
optimization levels
description 204
Oracle sources
pushdown optimization 213
partitioned 233
order
maintain in partitioned mapping 243
stable sort in partitioned mapping 244
output expression
configuring in a mapplet 88
output expression
how to configure 82
output file directory
multiple directories 236
optimizing 236
Outputs Binding
dialog box description 76
Outputs view
description 68
P
parallelism
decreasing for mapping 244
parameter files
application element 54
parameter files (continued)
creating 56
project element 53
purpose 52
running mappings with 52
sample 54
structure 52
XML schema definition 52
parameter instance value
setting 47
parameter sets
creating 64
overview 51
parameters
exporting to PowerCenter 179
flat file delimiter 45
how to configure 56
importing from PowerCenter 194
in expressions 42
in mapplets 47
in SQL statements 43
instance values 63
logical data objects 49
table names and resources 44
virtual table mappings 50
partition points
description 229
partitioning
Address Validator transformation 246
Aggregator transformation 241
cache 241
cache size 242
concurrent read 232
Decision transformation 241
decreasing for mapping 244
Expression transformation 241
flat file sources 232
flat file targets 235
IBM DB2 for LUW sources 233
IBM DB2 for LUW targets 239
Java transformation 241
Joiner transformation 241
maintain row order 243
maintain stable sort order 244
mappings 239
Match transformation 246
maximum parallelism 230
merged file targets 236
Oracle sources 233
Rank transformation 241
reject files 174
relational connection types 234, 240
SQL transformation 241
transformations 240
troubleshooting 248
partitioning restrictions
Aggregator transformation 241
Decision transformation 241
Expression transformation 241
Java transformation 241
Joiner transformation 241
numeric functions 243
Rank transformation 241
relational sources 234, 235
relational targets 240
SQL transformation 241
performance tuning
branch pruning optimization method 201
performance tuning (continued)
cost-based optimization method 202
dataship-join optimization method 202
early projection optimization method 200
early selection optimization method 201
optimization levels 204
optimization methods 200
predicate optimization method 201
pushdown optimization 207
semi-join optimization method 203
persisting mapping outputs
binding to task input 85
description 72
how to 83
rules and guidelines 73
using infacmd 73
pipeline stages
description 229
port attributes
propagating 19
port list parameters
description 46
port preview
port selector 110
port selector
creating 130
description 116
in dynamic expressions 113
selection rules 111
where used 112
ports
connection validation 30
linking 17
linking automatically 18
linking by name 18
linking by position 18
linking manually 18
linking rules and guidelines 19
propagated attributes by transformation 21
processing threads
mappings 229
project element
parameter files 53
pushdown optimization
expressions 214
Relational sources 211
SAP sources 214
functions 215
Greenplum sources 213
IBM DB2 sources 213
Microsoft SQL Server sources 213
nonrelational sources on z/OS 213
ODBC sources 213
operators 225
Oracle sources 213
overview 207
pushdown type 208
relational sources 213
SAP HANA sources 213
sources 211
Sybase ASE sources 213
pushdown optimization method
configuring pushdown 209
full pushdown 208
source pushdown 209
R
Rank transformation
\begin{itemize}
\item cache partitioning 241
\item multiple cache directories 242
\item partitioned 241
\end{itemize}
reject file
\begin{itemize}
\item column indicators 176
\item row indicators 175
\end{itemize}
reject files
\begin{itemize}
\item locating 174
\item partitioning 174
\item reading 174
\item targets 174
\item viewing 174
\end{itemize}
relational data objects
\begin{itemize}
\item partitioned 233, 239
\item reject files 174
\end{itemize}
relational sources
\begin{itemize}
\item partitioned 233
\item partitioning restrictions 234, 235
\item pushdown optimization 213
\end{itemize}
relational targets
\begin{itemize}
\item partitioned 239
\item partitioning restrictions 240
\item reject files 174
\end{itemize}
row indicators
\begin{itemize}
\item reject file 175
\end{itemize}
runtime links
\begin{itemize}
\item creating 136
\item link policy 116
\item overview 116
\end{itemize}
S
SAP HANA sources
\begin{itemize}
\item pushdown optimization 213
\end{itemize}
SAP sources
\begin{itemize}
\item pushdown optimization 214
\end{itemize}
scope
\begin{itemize}
\item port selector 111
\end{itemize}
segments
\begin{itemize}
\item copying 31
\item selection criteria
\item port selector 111
\item selection rules
\item port selectors 111
\item semi-join optimization
\item description 203
\item sort list parameters
\item description 46
\item sort order
\item maintain in partitioned mapping 243
\end{itemize}
Sorter transformation
\begin{itemize}
\item cache partitioning 241
\item sources
\item partitioned flat file 232
\item partitioned relational 233
\item specify by
\item value or parameter 43
\end{itemize}
SQL statements
\begin{itemize}
\item parameters in 43
\end{itemize}
SQL transformation
\begin{itemize}
\item disable partitioning 243
\item partitioned 241
\item stable sort order
\item maintain in partitioned mapping 244
\item stop on errors
\item mapping property 23
\item string parameters
\item precision limit 37
\item suggested parallelism
\item transformations 245
\end{itemize}
Sybase ASE sources
\begin{itemize}
\item pushdown optimization 213
\end{itemize}
system-defined parameters
\begin{itemize}
\item importing 195
\end{itemize}
T
target commit interval
\begin{itemize}
\item mapping property 23
\end{itemize}
target load order constraints
\begin{itemize}
\item creating 28
\item description 26
\item example 28
\item Load Order tab 26
\item rules and guidelines 27
\end{itemize}
targets
\begin{itemize}
\item merge file 236
\item partitioned flat file 235
\item partitioned relational 239
\end{itemize}
threads
\begin{itemize}
\item processing mappings 229
\end{itemize}
transformations
\begin{itemize}
\item partitioned 240
\item suggested parallelism 245
\end{itemize}
troubleshooting
\begin{itemize}
\item dynamic mappings 138
\item exporting objects to PowerCenter 184
\end{itemize}
U
updateMappingPersistedOutputs
\begin{itemize}
\item description 73
\end{itemize}
V
virtual table mappings
\begin{itemize}
\item configuring parameters 50
\end{itemize}
W
workflow variables
\begin{itemize}
\item binding mapping outputs to 86
\end{itemize}