Informatica®

Informatica®

10.5.10

Developer Mapping Guide

Informatica Developer Mapping Guide
10.5.10
February 2026

© Copyright Informatica LLC 2014, 2025

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. 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.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, the Informatica logo, PowerCenter, and PowerExchange are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. 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. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2026-02-19

Table of Contents

Preface 12

Informatica Resources 12
- Informatica Network 12
- Informatica Knowledge Base 12
- Informatica Documentation 12
- Informatica Product Availability Matrices 13
- Informatica Velocity 13
- Informatica Marketplace 13
- Informatica Global Customer Support 13

Chapter 1: Mappings 14

Mappings Overview 14

Mapping Components 15
- Data Object Operations 15
- Transformations 20
- Mapplets 20
- Segments 21
- Copying a Segment 21

Views 21

Mapping Validation 22
- Connection Validation 23
- Expression Validation 23
- Object Validation 23
- Parameter Validation 23
- Synchronization Validation 23

Mapping Run-time Properties 24
- Validation Environment 24
- Execution Environment 25
- Reject File Directory 25
- Maximum Parallelism 26
- Target Commit Interval 26
- Stop on Errors 27
- Mapping Impersonation User Name 27
- Suggested Parallelism 27

Target Load Order Constraints 28
- Constraints with Insert and Delete Rows 28
- Target Load Order Rules and Guidelines 29
- Target Load Order Example 29

Mapping Configurations 31
- Mapping Configuration Properties 32

How to Configure the Default Mapping Configuration. . . . 34
How to Create a Reusable Mapping Configuration. . . . 34
How to Create and Use a Custom Mapping Configuration. . . . 34
Advanced Mapping Options. . . . 35
How to Run a Mapping Using Advanced Options. . . . 36
How to Develop a Mapping. . . . 38
Creating a Mapping. . . . 38
Adding Objects to a Mapping. . . . 38
Connecting Mapping Objects. . . . 39
Creating Target Load Order Constraints. . . . 39
Validating a Mapping. . . . 40
Running a Mapping. . . . 41

Chapter 2: Mapplets. . . . 42

Mapplets Overview. . . . 42
Mapplet Types. . . . 42
Mapplet Input and Output. . . . 43
Mapplet Input. . . . 43
Mapplet Output. . . . 44
Generated Mapplets. . . . 44
Generated Mapplet Rules and Guidelines. . . . 44
Generating a Mapplet. . . . 44
Rule Specifications and Mapplets. . . . 46
Rule Specification Properties. . . . 46
Creating a Mapplet. . . . 47
Mapplet Validation. . . . 47
Validating a Mapplet. . . . 48
Mapplet as a Rule Validation. . . . 48

Chapter 3: Mapping Parameters. . . . 49

Mapping Parameters Overview. . . . 49
System Parameters. . . . 50
User-Defined Parameters. . . . 51
Date/Time Parameters. . . . 52
Where to Create User-Defined Parameters. . . . 53
Parameters in Mappings. . . . 53
Parameter Instance Value. . . . 54
Parameters in Mapplets. . . . 55
Parameter Instance Values in Mapplets. . . . 55
Mapplet Parameters in Mappings. . . . 56
Parameters in Mapplets Example. . . . 56
Parameters in Logical Data Objects. . . . 57
Parameters in Virtual Table Mappings. . . . 58

Parameter Sets . . . 59
Parameter Files. . . . 60
Parameter File Structure . . . 60
Project Element. . . . 61
Application Element. . . . 62
Rules and Guidelines for Parameter Files. . . . 62
Sample Parameter File. . . . 63
Export a Parameter File. . . . 64
Creating a Parameter File from infacmd ms ListMappingParams . . . 64
Parameter Hierarchy. . . . 65
Overriding Parameters Using the Parameter Hierarchy. . . . 65
Binding Parameters to Override the Parameters at Run Time. . . . 66
Use Cases for Overriding Parameters in Mappings. . . . 66
How to Configure Parameters. . . . 68
Creating a Parameter for a Transformation Property. . . . 69
Creating a Parameter in an Expression. . . . 71
Expose Transformation Parameters as Mapping Parameters. . . . 74
Setting the Parameter Instance Value. . . . 75
Creating a Parameter Set. . . . 76
How to Run a Mapping with Parameters. . . . 78
How to Run a Mapping with Parameters from the Developer Tool. . . . 78
How to Run a Mapping with Parameters from the Command Line. . . . 81

Chapter 4: Where to Assign Parameters. . . . 82

Where to Assign Parameters Overview. . . . 82
Parameters for Compression Formats. . . . 86
Parameters in Custom Queries for Hive Sources. . . . 87
Parameters in Custom Queries for Relational Sources. . . . 88
Parameters in Expressions. . . . 89
Expression Parameters. . . . 91
Expression Parameter Examples. . . . 92
Example. Expression Parameters in a Filter Condition. . . . 92
Example. Expression Parameters in an Expression Transformation. . . . 93
Example. Nesting Parameters in Port Expressions. . . . 94
Parameters for Fields and Property Values. . . . 95
Parameters for Relational Table Resources. . . . 96
Parameters in SQL Statements. . . . 96
Parameters in Filter and Join Conditions. . . . 97
String Parameters in SQL Statements. . . . 98
Tips for Using Parameters in SQL Statements. . . . 99
Parameters for Port Lists. . . . 99

Chapter 5: Mapping Outputs 100

Mapping Outputs Overview 100
User-Defined Mapping Outputs 101
Outputs View 101
Mapping Output Expression 102
System-Defined Mapping Outputs 103
Persist and Bind Mapping Outputs in Deployed Mappings 104
How to Configure and Bind Mapping Outputs 104
Persisted Outputs Maintenance 108
Rules and Guidelines for Persisted Mapping Outputs and Deployment 109
Mapping Outputs In Mapplets 109
Bind Mapplet Outputs to Mapping Outputs 111
Mapping Outputs in Logical Data Objects 113
How to Bind Mapplet Outputs to Mapping Outputs 113
Defining Mapplet Outputs 114
Configuring a Mapping Output Expression in a Mapplet 114
Binding Outputs from a Mapplet to Mapping Outputs 115

Chapter 6: Generate a Mapping from an SQL Query 117

Generate a Mapping from an SQL Query Overview 117
Example of Generated Mapping from an SQL Query 117
SQL Syntax to Generate a Mapping 118
Correlated Subqueries 118
Function Support in Queries that Generate a Mapping 119
Generate a Mapping from an SQL Query with an Unsupported Function 119
INSERT, UPDATE and DELETE Syntax 120
Rules and Guidelines for INSERT, UPDATE, and DELETE Statements 120
Generating a Mapping or Logical Data Object from an SQL Query 121
Generate a Mapping from an SQL Statement 121
Create an SQL Statement 122
Paste or Import the SQL Statement to the Developer Tool 122
Complete Mapping Development 123

Chapter 7: Dynamic Mappings 124

Dynamic Mappings Overview 124
Dynamic Mapping Configuration 125
Dynamic Data Sources 125
Dynamic Mapping Ports and Links 126
Dynamic Mapping Rules 127
Parameters in Dynamic Mappings 127
Dynamic Sources 128
Get Columns from the Data Source 129

Assign a Parameter to a Flat File Name. . . . 130
Assign a Parameter to Relational Source Properties. . . . 131
Assign a Parameter to the Source Data Object. . . . 131
Dynamic Targets. . . . 132
Get Columns from the Data Source 134
Define Targets Based on the Mapping Flow. . . . 134
Define Targets Based on the Data Object. . . . 135
Define a Target Schema Strategy. . . . 135
Assign a Parameter to Relational Target Properties 137
Assign a Parameter to the Target Data Object. . . . 137
Rules and Guidelines for Dynamic Targets. . . . 138
Dynamic Ports and Generated Ports. . . . 138
Dynamic and Generated Port Configuration. . . . 139
Rules and Guidelines for Dynamic and Generated Ports. . . . 140
Dynamic Expressions. . . . 140
Parameterizing the Dynamic Expression. . . . 141
Input Rules. . . . 142
Input Rule Configuration. . . . 142
Include or Exclude Ports. . . . 143
Include All Remaining Ports. . . . 144
Rename Generated Ports. . . . 145
Restore Source Port Names. . . . 147
Reorder Generated Ports. . . . 148
Selection Rules and Port Selectors. . . . 151
Port Selector Configuration. . . . 151
Selection Rules. . . . 152
Example - Selection Rules and Port Selectors. . . . 153
Design-time Links 153
Link Resolution. . . . 155
Run-time Links. . . . 155
Run-time Link Configuration. . . . 156
Example - Run-time Links. . . . 157
Troubleshooting Dynamic Mappings. . . . 158

Chapter 8: How to Develop and Run a Dynamic Mapping. . . . 160

Developing and Running Dynamic Mappings. . . . 160
Configuring a Dynamic Source. . . . 161
Using a Parameter as a Source for a Dynamic Mapping. . . . 162
Configuring Sources to Get Metadata Changes at Run Time. . . . 162
Creating a Dynamic Port. . . . 163
Configuring Dynamic Ports Using Input Rules. . . . 164
Step 1. Open the Input Rules Dialog box. . . . 165
Step 2. Define Input Rules. . . . 165

Step 2a. Choose the Operator and Selection Criteria. . . . 165
Step 2b. Configure the Name Selection Criteria Details. . . . 166
Step 2c. Configure the Type Selection Criteria Details. . . . 166
Step 2d. Configure the Pattern Selection Criteria Details. . . . 167
Step 3. Rename the Generated Ports. . . . 167
Step 4. Reorder the Generated Ports. . . . 168
Step 5. Verify the Dynamic Port Configuration. . . . 168
Creating a Port Selector. . . . 168
Creating a Dynamic Expression 169
Configuring a Dynamic Target. . . . 171
Using a Parameter as a Target for a Dynamic Mapping. . . . 172
Getting Target Object Columns from the Data Source at Run-Time. . . . 173
Defining a DDL Query to Create or Replace the Target at Run Time. . . . 173
Defining Write Transformation Ports. . . . 174
Creating and Configuring a Run-time Link. . . . 176
Validating a Dynamic Mapping. . . . 178
Validating Dynamic Sources and Targets. . . . 178
Running a Dynamic Mapping. . . . 179

Chapter 9: Dynamic Mapping Use Cases. . . . 180

Use Case: Dynamic Mapping for Metadata Changes in Relational Sources. . . . 180
Source Tables. . . . 180
Target Table. . . . 181
Dynamic Mapping. . . . 181
Step 1. Configure the Read Transformations. . . . 182
Step 2. Configure the Joiner Transformation. . . . 183
Step 3. Configure the Aggregator Transformation. . . . 184
Step 4. Configure the Write Transformation. . . . 187
Step 5. Create and Configure a Run-time Link. . . . 188
Step 6. Validate and Run the Mapping. . . . 188
Step 7. Run the Mapping after Changes to the Source Schema. . . . 189
Use Case: Reuse Dynamic Mapping for Different Sources and Targets. . . . 191
Source Files. . . . 191
Target Files. . . . 193
Dynamic Mapping. . . . 193
Step 1. Configure the Read_Customer_FF Read Transformation. . . . 194
Step 2. Configure the Exp_TRIM Expression Transformation. . . . 195
Step 3. Configure the Exp_Output Expression Transformation. . . . 199
Step 4. Configure the Write_customerTrim_FF Write Transformation. . . . 201
Step 5. Validate and Save the Mapping. . . . 202
Step 6. Run the Dynamic Mapping Against Different Sources and Targets. . . . 203

Chapter 10: Mapping Administration. 206

Mapping Administration Overview. 206
Viewing Properties for a Mapping Job. 207
Viewing Summary Statistics for a Mapping Job. 207
Viewing Detailed Statistics for a Mapping Job. 207
Viewing Logs for a Mapping Job. 208
Reissuing a Deployed Mapping Job. 208
Canceling a Mapping Job. 209
Reject Files. 209
Location of Reject Files. 209
Content of Reject Files. 209

Chapter 11: Export to PowerCenter. 212

Export to PowerCenter Overview. 212
PowerCenter Release Compatibility. 213
Setting the Compatibility Level. 213
Mapplet Export. 213
Mappings with Parameters Export. 214
Export to PowerCenter Options. 214
Exporting an Object to PowerCenter. 215
Export Restrictions. 216
Rules and Guidelines for Exporting to PowerCenter. 218
Troubleshooting Exporting to PowerCenter. 219

Chapter 12: Import From PowerCenter. 220

Import from PowerCenter Overview. 220
Override Properties. 221
Conflict Resolution. 223
Import Summary. 224
Data Type Conversion. 224
Transformation Conversion. 224
Transformation Property Restrictions. 225
Parameter Conversion. 231
Rules and Guidelines for Importing Mappings with Parameters 231
Rules and Guidelines for Importing Mapping with Variables. 232
Rules and Guidelines for Importing Overrides. 232
System Parameter Conversion. 232
PowerCenter Repository Connection Properties. 233
Connection Assignments. 234
Importing an Object from PowerCenter. 235
Import Restrictions. 236
Source and Target Restrictions. 236

Transformation Restrictions. . . . 237
Mapping Restrictions. . . . 237
Functions Restrictions. . . . 238
Mapping Variables Restrictions. . . . 238
Session Properties Restrictions. . . . 238
Workflow Restrictions. . . . 238
Task Restrictions. . . . 239
Import Performance. . . . 239

Chapter 13: Performance Tuning. . . . 240

Performance Tuning Overview. . . . 240
Optimization Methods. . . . 241
Early Projection Optimization Method. . . . 241
Early Selection Optimization Method. . . . 242
Branch Pruning Optimization Method. . . . 242
Predicate Optimization Method. . . . 242
Cost-Based Optimization Method. . . . 243
Dataship-Join Optimization Method. . . . 243
Semi-Join Optimization Method. . . . 244
Viewing an Optimized Mapping. . . . 245
Optimizer Levels. . . . 245
Setting the Optimizer Level for a Developer Tool Mapping. . . . 246
Setting the Optimizer Level for a Deployed Mapping. . . . 247

Chapter 14: Pushdown Optimization. . . . 248

Pushdown Optimization Overview. . . . 248
Pushdown Types. . . . 249
Full Pushdown Optimization. . . . 249
Source Pushdown. . . . 250
Configuring Pushdown. . . . 250
Transformation Pushdown Logic. . . . 251
Pushdown Optimization to Sources. . . . 252
Pushdown Optimization to Relational Sources. . . . 252
Pushdown Optimization to Native Sources. . . . 254
Pushdown Optimization to PowerExchange Nonrelational Sources. . . . 254
Pushdown Optimization to ODBC Sources. . . . 254
Pushdown Optimization to SAP Sources. . . . 255
Pushdown Compatibility. . . . 255
Operator Compatibility. . . . 255
Functions for IBM DB2. . . . 256
Functions for Cloud Data Warehouse Applications. . . . 259
Functions for Data Warehouse Applications. . . . 262
Functions for Enterprise Applications. . . . 265

Comparing the Output of the Data Integration Service and Sources. 267

Chapter 15: Partitioned Mappings. 269

Partitioned Mappings Overview. 269
One Thread for Each Pipeline Stage. 270
Multiple Threads for Each Pipeline Stage. 271
Partitioned Flat File Sources. 273
Concurrent Read Partitioning. 273
Partitioned Relational Sources. 274
Relational Connection Types for Partitioning. 275
SQL Queries for Partitioned Relational Sources. 275
Rules and Guidelines for Relational Source Partitions. 276
Partitioned Flat File Targets. 276
Optimize Output File Directories for Partitioned File Targets. 277
Merge Options for Partitioned File Targets. 277
Commands for Partitioned File Targets. 279
Partitioned Relational Targets. 280
Relational Connection Types for Partitioning. 281
Rules and Guidelines for Relational Target Partitions. 281
Partitioned Transformations. 281
Restrictions for Partitioned Transformations. 282
Cache Partitioning for Transformations. 282
Disable Partitioning for a Transformation. 284
Maintain Order in a Partitioned Mapping. 284
Maintain a Stable Sort. 285
Override the Maximum Parallelism for a Mapping. 285
Suggested Parallelism for a Transformation. 286
Execution Instances for Address Validator and Match Transformations. 287
Overriding the Maximum Parallelism Value. 288
Troubleshooting Partitioned Mappings. 289

Chapter 16: Developer Tool Naming Conventions. 290

Transformation Naming Conventions. 290
Object Type Naming Conventions. 292
Workflow Object Naming Conventions. 292

Index. 294

Preface

Use the *Informatica Developer Mapping Guide* to learn how to develop, run, and administer mappings. Understand mapping concepts, how to create flexibility through mapping parameters and dynamic mappings, and how to optimize mappings through tuning and partitioning.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Mappings

This chapter includes the following topics:

- Mappings Overview, 14
- Mapping Components, 15
- Views, 21
- Mapping Validation, 22
- Mapping Run-time Properties, 24
- Target Load Order Constraints, 28
- Mapping Configurations, 31
- Advanced Mapping Options, 35
- How to Develop a Mapping, 38

Mappings Overview

A mapping is a set of input and output objects that represent the data flow between sources and targets. They are 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.

You can 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. You can include a mapping with physical data objects as the input and output in a mapping task in a workflow.

The type of input and output object you include in a mapping determines the type of mapping. You can create the following types of mappings in the Developer tool:

Logical data object mapping

Links a logical data object to one or more physical data objects. A logical data object mapping helps you to integrate data from multiple sources and formats into a standardized view.

Operation mapping

Has an operation as the mapping input, output, or both. An operation mapping performs the web service operation for the web service client.

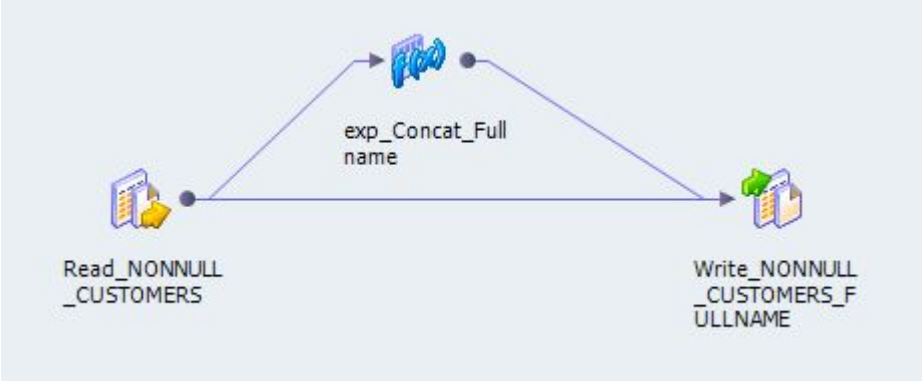
Virtual table mapping

Has a virtual table as the mapping output. A virtual table mapping defines the virtual data flow between sources and a virtual table in an SQL data service. Use a virtual table mapping to transform the data.

Dynamic mapping

A mapping in which you can change the sources, targets, and transformation logic at run time based on parameters and rules that you define. Use a dynamic mapping to manage frequent schema or metadata changes.

The following image shows an example of a mapping:



Mapping Components

Mapping components determine the data flow between sources and targets.

Every mapping must contain an input object, which reads data from a mapping component or file. Every mapping must also contain an output object, which writes data to a mapping component or file.

A mapping can also contain the following components:

Data object operations

Repository objects that contain properties required to perform certain run-time operations on sources or targets. Required for some PowerExchange adapter data sources.

Transformations

Modify data before writing it to targets. Use different transformation objects to perform different functions.

Mapplets

Reusable objects containing a set of transformations that you can use in multiple mappings.

Segments

Consist of one or more objects in a mapping, mapplet, rule, or virtual stored procedure.

Data Object Operations

A data object operation is a repository object that contains properties required to perform certain run-time operations on sources or targets. Some PowerExchange adapter data sources have a complex structure that do not allow the Developer tool to import all properties that are necessary at mapping run-time.

For example, you import a PowerExchange Microsoft Dynamic CRM source, but it does not import with precision and scale. In addition, you might need to perform specific run-time operations on Microsoft

Dynamics CRM, or you might want to control the maximum number of records that the Data Integration Service can read in one batch. You can configure these properties in the data object read operation.

When you import data from a PowerExchange adapter data source, the data stored in the source's complex structure might also not be compatible with transformations in the Developer tool. You can use a data object read operation to convert the data types native to the source into transformation data types that the Developer tool can use in the mapping workflow.

When you write data to a target in a PowerExchange adapter data source with a complex structure, you might need to similarly use a data object write operation to convert data from the transformation data types back to the data types native to the data source.

If you import a physical data object from a resource, you can create data object operations for the physical data object based on the same resource. The resource is the part of the data object from which you want to read data. For example, the resource for a database can be a table or a view.

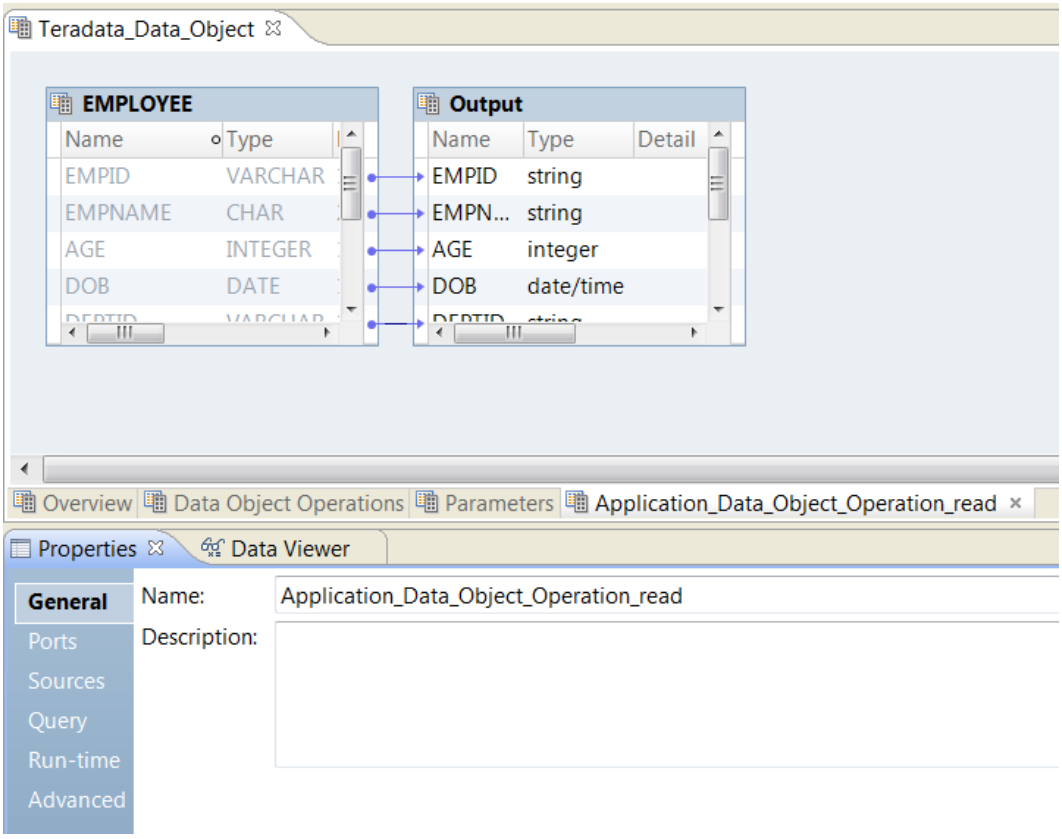
After you create the physical data object and the required data object operations, you can view the physical data object in the object editor. The object editor view includes a tab where you can configure the data object operation properties. A data object can have multiple read and write operations.

Data Object Read Operation

A data object read operation is associated with a source data object. You can create a data object read operation and configure the data object read operation properties in the object editor.

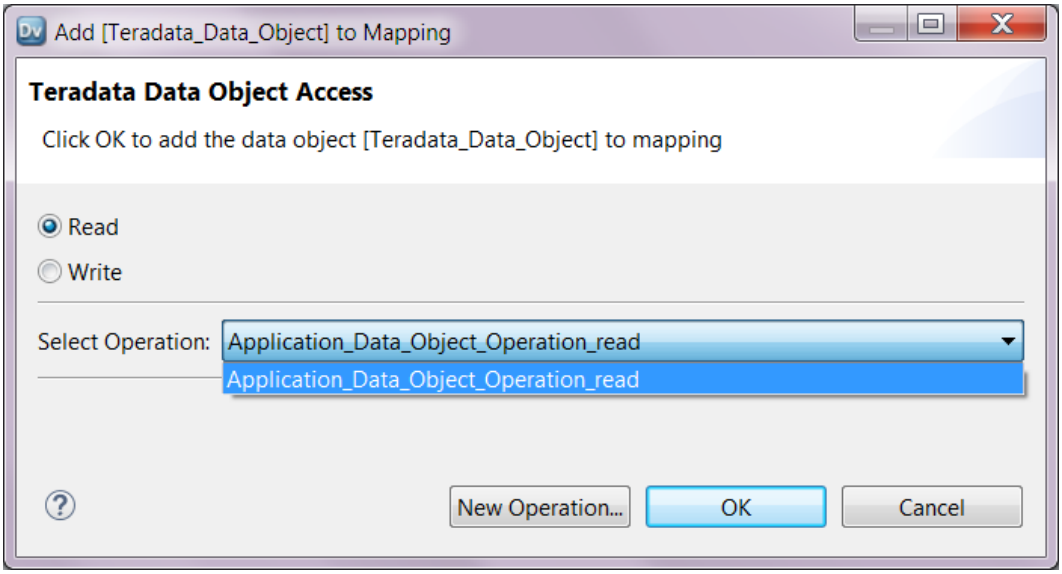
When you view the data object read operation in the object editor, the object editor displays a source object and an output object. The source and output objects comprise the data object read operation. You edit the advanced and run-time properties in the output object.

The following image shows an example of a data object read operation for a Teradata data object where the source object is `EMPLOYEE` and the output object is `Output`:



After you create the data object read operation for the physical data object, you can create a Read transformation to add the physical data object as a source in the mapping workflow. When you add the physical data object to the mapping, you can specify the Read transformation and the data object read operation that you want to use.

The following image shows the wizard that appears when you add the Teradata data object to the mapping:



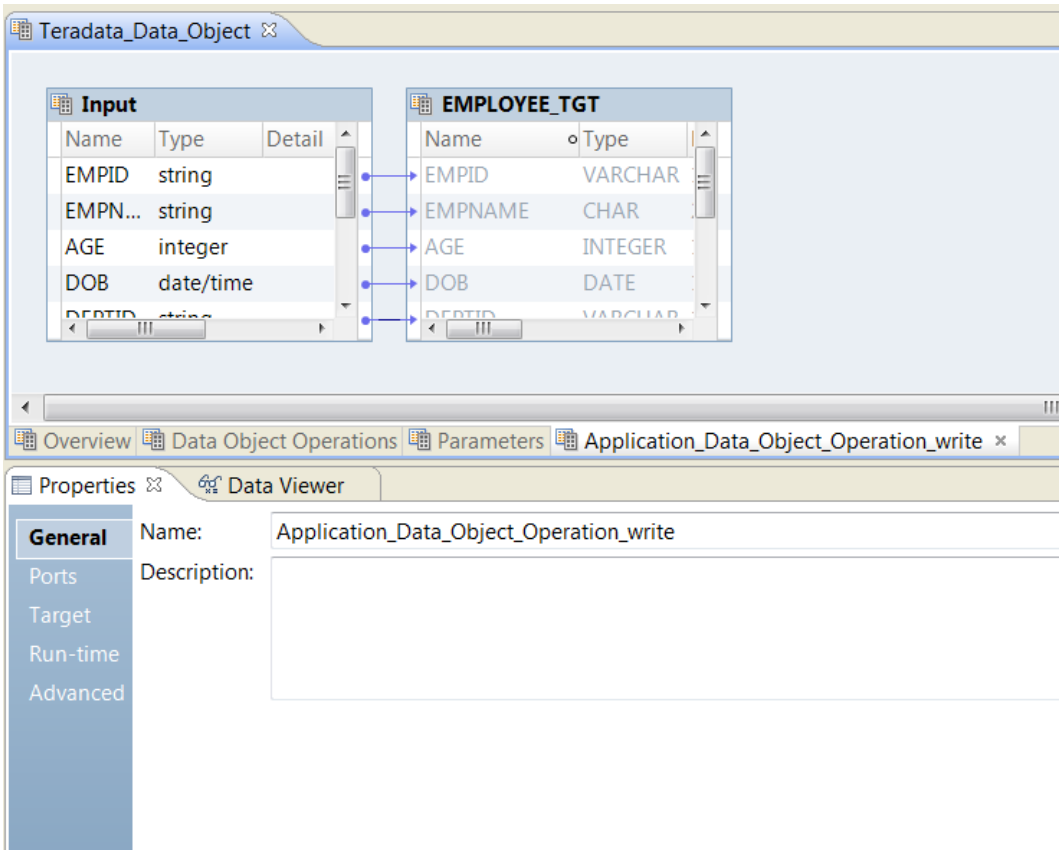
The Read transformation uses the properties that you configured in the output object of the data object read operation.

Data Object Write Operation

A data object write operation is associated with a target data object. You can create a data object write operation and configure the data object write operation properties in the object editor.

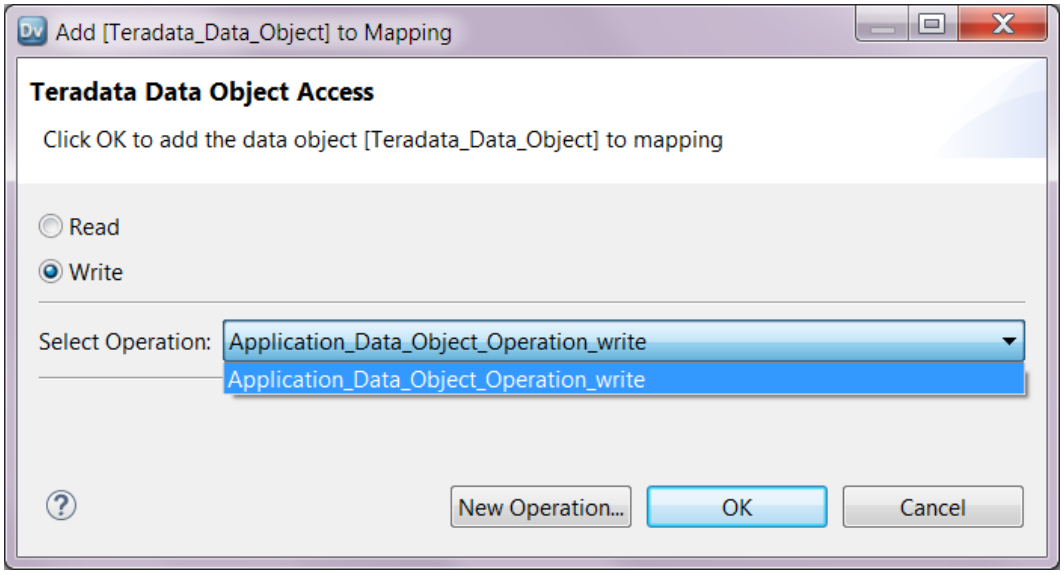
When you view the data object write operation in the object editor, the object editor displays an input object and a target object. The input and target objects comprise the data object write operation. You edit the advanced and run-time properties in the input object.

The following image shows an example of a data object write operation for a Teradata data object where the input object is `Input` and the target object is `EMPLOYEE_TGT`:



After you create the data object write operation for the physical data object, you can create a Write transformation to add the physical data object as a target in the mapping workflow. When you add the physical data object to the mapping, you can specify the Write transformation and the data object write operation that you want to use.

The following image shows the wizard that appears when you add the Teradata data object to the mapping:



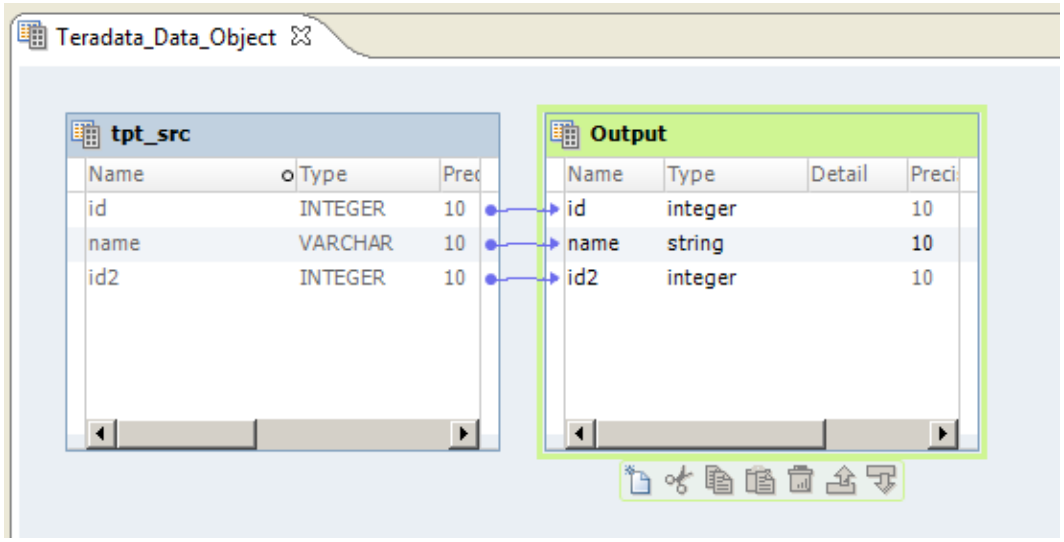
The Write transformation uses the properties that you configured in the input object of the data object write operation.

Teradata Data Object Example

A Teradata data object represents metadata based on a Teradata resource.

When you configure a data object read operation for the Teradata data object and view it in the object editor, you can see a source object and an output object. The output object displays the metadata from the Teradata resource.

The following image shows a data object read operation for the Teradata data object:



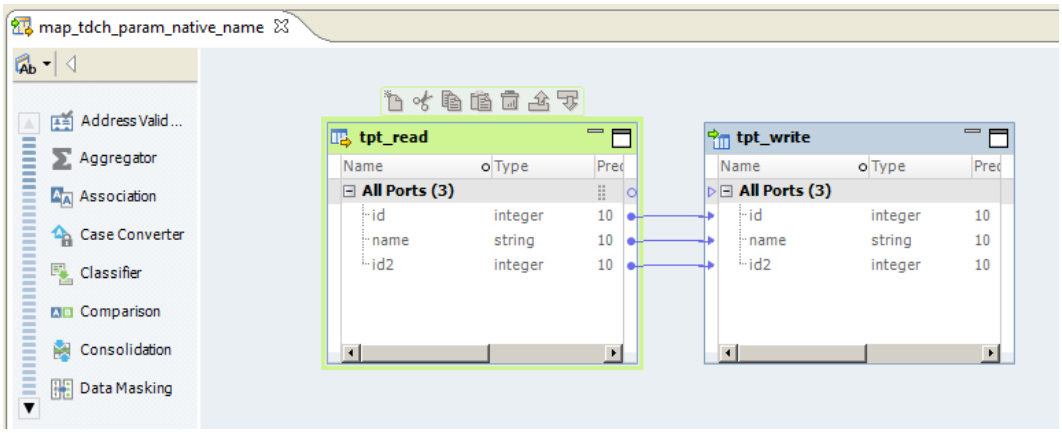
The source object is `tpt_src` and the output object is `Output`. The source and output objects comprise the data object read operation.

You can look at the metadata in the output object and see that the native data types in `tpt_src` are converted into transformation data types in the output object. For example, the native data type VARCHAR for the **name** port is converted to the transformation data type string.

To use the Teradata data object in a mapping, you can create a Read transformation. When you add the Read transformation to the mapping, you select the Teradata data object and the data object read operation that you previously configured.

Because the Read transformation needs to access the transformation data types stored in the output object of the data object read operation, the Read transformation is created based on the object operation, not the Teradata data object. The Read transformation uses the same properties that you configured for the data object read operation.

The following image highlights the Read transformation `tpt_read` in the mapping editor:



Notice that the metadata in the Read transformation and the output object of the data object read operation are the same. For example, the metadata in the **name** port uses the transformation data type string instead of the native data type VARCHAR.

Transformations

A transformation is an object that generates, modifies, or passes data.

Informatica Developer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data. Transformations in a mapping represent the operations that the Data Integration Service performs on the data. Data passes through transformation ports that you link in a mapping or mapplet.

Transformations can be active or passive. Transformations can be connected to the data flow, or they can be unconnected. For more information about transformations, see the *Developer Transformation Guide*.

Mapplets

A mapplet is a reusable object containing a set of transformations that you can use in multiple mappings.

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 one time in a mapping or mapplet. You can create a mapplet manually. You can also generate a mapplet from a segment within a mapping or mapplet.

For more information about mapplets, see Chapter 2, “Mapplets” on page 42.

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 across folders or projects. Consider the following rules and guidelines when copying segments:

- The Developer tool reuses dependencies when possible. If it cannot reuse the dependencies, 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.

You can also generate a mapplet from a segment in a mapping or mapplet. You might want to generate a mapplet when a mapping or mapplet contains a connected transformation flow that you want to reuse. For more information about generating mapplets, see "Generating a Mapplet" on page 44.

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.

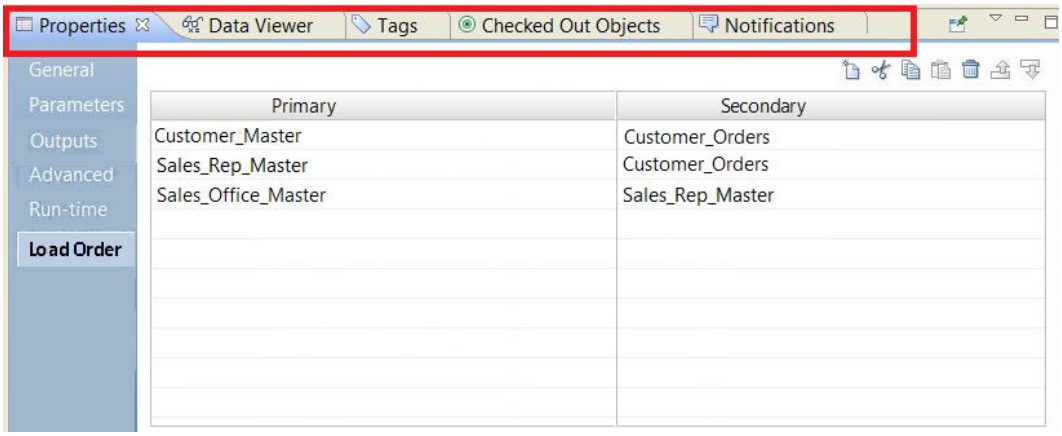
1. Open the object that contains the segment that you want to copy.
2. Select a segment by highlighting each object you want to copy.

   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**.

Views

Different views become available when you click inside the editor. A view is a workbench part that can navigate an information hierarchy or display object properties. You can switch between these views to modify properties or other details in the editor. You can also use these views to select objects that you want to appear in the editor.

The following image shows the different views in Informatica Developer:



Switch between the following views to perform different tasks:

Properties

Configure general mapping properties, including mapping name, run-time properties, and load order constraints.

Data Viewer

Preview the data and view the mapping output for each transformation. You can also export data in the **Data Viewer** view.

Tags

Create a tag to add metadata, assign a tag to an object, and view all tags assigned to an object.

Checked Out Objects

View objects that you have checked out.

Notifications

Set up and configure global settings for scorecard notifications. You can also select recipients in the Informatica domain to receive notifications during a workflow.

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 performs the following types of validation:

- Connection validation
- Expression validation
- Object validation
- Parameter validation
- Synchronization validation

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 performs 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.
- Data types between ports are compatible. If you change a port data type to one that is incompatible with the port it is connected to, the Developer tool generates an error and invalidates the mapping. However, you can change the data type 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 do 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.

Parameter Validation

To validate mapping parameters in the Developer tool, view the mapping with resolved parameters and validate the mapping.

When you resolve mapping parameters, the Developer tool generates a run-time instance of the mapping that shows the resolved parameters at run-time. Validate the run-time instance of the mapping to validate the mapping parameters. You can validate mapping parameters that use the default parameter values in the mapping, a parameter set, or a parameter file.

Synchronization Validation

You can validate run-time synchronization of dynamic sources and targets.

If you configure a dynamic mapping that uses data sources that are set to synchronize at run time, you can preview the run-time instance of the mapping to validate the synchronization. To preview the run-time instance of the mapping, view the mapping with resolved parameters. When you view the mapping with resolved parameters, the Developer tool generates a run-time instance of the mapping that reflects run-time changes in dynamic sources and targets. Validate the run-time mapping to validate data source synchronization.

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 Environment
- Execution Environment
- Reject File Directory
- Maximum Parallelism
- Target Commit Interval
- Stop On Errors
- Mapping Impersonation User Name
- Suggested Parallelism

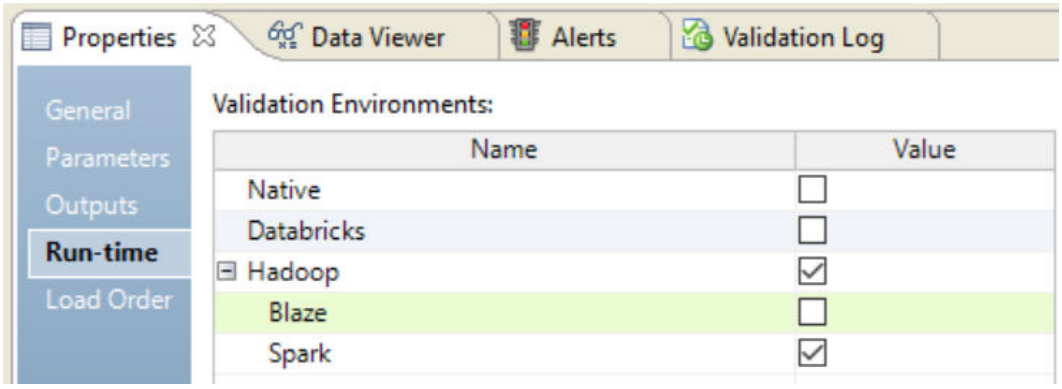
Validation Environment

The validation environment indicates whether the Developer tool validates the mapping definition for the native or non-native execution environment. 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 non-native environment. When you run a mapping in the non-native environment, the Data Integration Service pushes the mapping execution to the compute cluster through a cluster connection. The compute cluster processes the mapping.

When you choose the Hadoop execution environment, you can select the Blaze or Spark engine to process the mapping.

The following image shows the validation environment:



Choose native, Hadoop, and Databricks environments in the following situations:

- You want to test the mapping in the native environment before you run the mapping in a non-native environment.
- You want to define the execution environment value in a parameter when you run the mapping.

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

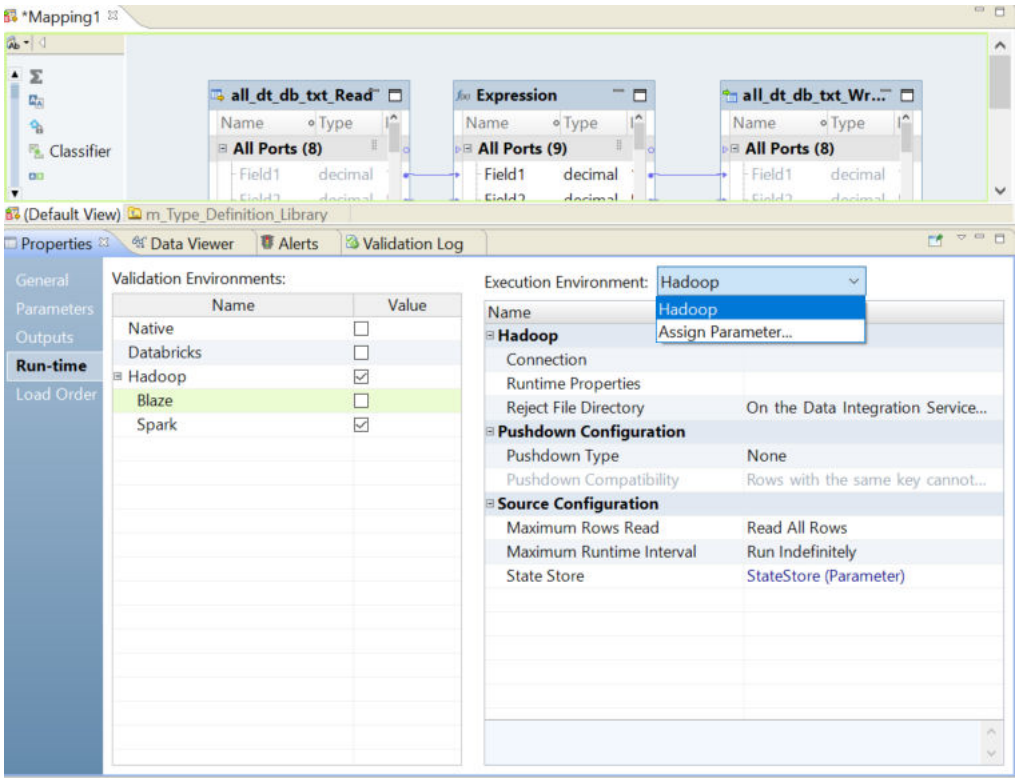
Note: If you run the mapping in the Hadoop environment, Informatica recommends choosing the Spark engine for new mapping development.

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. When you run a mapping in the non-native environment, the Data Integration Service pushes the mapping execution to the compute cluster through a cluster connection. The compute cluster processes the mapping.

You can use a mapping parameter to indicate the execution environment. Configure a string parameter. 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:



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

Reject File Directory

If you run mappings in the Hadoop environment, you can choose where to store the reject files if the Hadoop connection is configured with a reject file directory. The Blaze engine can write reject files to the Hadoop environment for flat file, HDFS, and Hive targets. The Spark engine can write reject files to the Hadoop environment for flat file and HDFS targets.

You can write reject files to the Data Integration Service machine or to the Hadoop cluster. Or, you can defer to the Hadoop connection configuration.

Choose one of the following options:

- On the Data Integration Service machine. The Data Integration Service stores the reject files based on the RejectDir system parameter.

- On the Hadoop Cluster. The reject files are moved to the reject directory configured in the Hadoop connection. If the directory is not configured, the mapping will fail.
- Defer to the Hadoop Connection. The reject files are moved based on whether the reject directory is enabled in the Hadoop connection properties. If the reject directory is enabled, the reject files are moved to the reject directory configured in the Hadoop connection. Otherwise, the Data Integration Service stores the reject files based on the RejectDir system parameter.

If you configure the mapping run-time properties to defer to the Hadoop connection, the reject files for all mappings with this configuration are moved based on whether you choose to write reject files to Hadoop for the active Hadoop connection. You do not need to change the mapping run-time properties manually to change the reject file directory.

For example, if the reject files are currently moved to the Data Integration Service machine and you want to move them to the directory configured in the Hadoop connection, edit the Hadoop connection properties to write reject files to Hadoop. The reject files of all mappings that are configured to defer to the Hadoop connection are moved to the configured directory.

You might also want to choose to defer to the Hadoop connection when the connection is parameterized to alternate between multiple Hadoop connections. For example, the parameter might alternate between one Hadoop connection that is configured to move reject files to the Data Integration Service machine and another Hadoop connection that is configured to move reject files to the directory configured in the Hadoop connection. If you choose to defer to the Hadoop connection, the reject files are moved depending on the active Hadoop connection in the connection parameter.

You cannot create the reject file directory at run time. Depending on the option that you choose, the configured reject file directory must exist on the Data Integration Service machine or in the Hadoop environment before you can store reject files.

Maximum Parallelism

Maximum parallelism is valid for the native execution environment. Maximum parallelism refers to the 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 one 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 Chapter 15, “Partitioned Mappings” on page 269.

Target Commit Interval

The target commit interval refers to the number of rows that 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

This function 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

A mapping impersonation user name is valid for the native and Hadoop execution environment. Use mapping impersonation 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.
- Host name is the name or IP address of the Hadoop service.
- YOUR-REALM is the Kerberos realm.

You can only use the following special characters as delimiters: '/' and '@'

Suggested Parallelism

Suggested parallelism is valid for the native execution environment when the Maximum Parallelism property is assigned to a value greater than one or to a parameter. Suggested number of parallel threads that process the transformation pipeline stage.

When you define a suggested parallelism value for a 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.

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.

Target load order for flat files.

You can configure a target load order constraint for a mapping that loads rows into multiple flat file targets. Configure the target load order to load the secondary flat file after the primary flat file.

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.

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 ignore 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 as 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.
- 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.

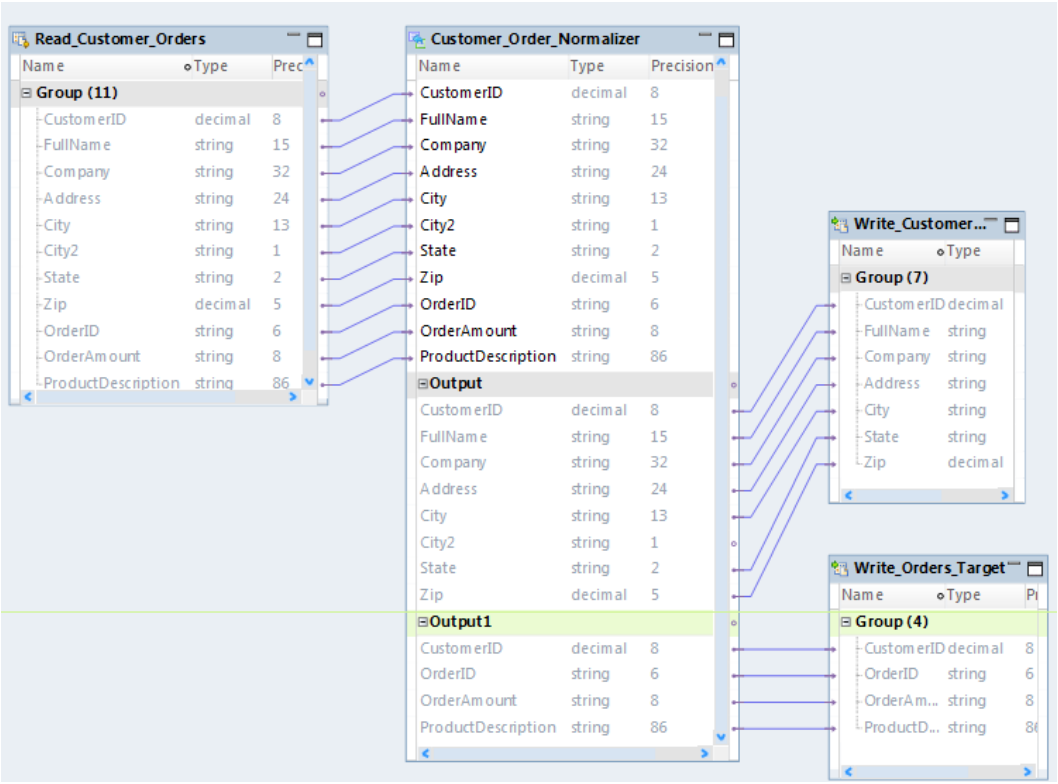
WARNING: Informatica strongly advises against using the staging files or the data within the files. Informatica is not responsible for corrupted data that is caused by customer alteration of the staging files or the data in the tables. The structure of the staging files might change between Informatica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

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 the it completes loading the customer information to the target.

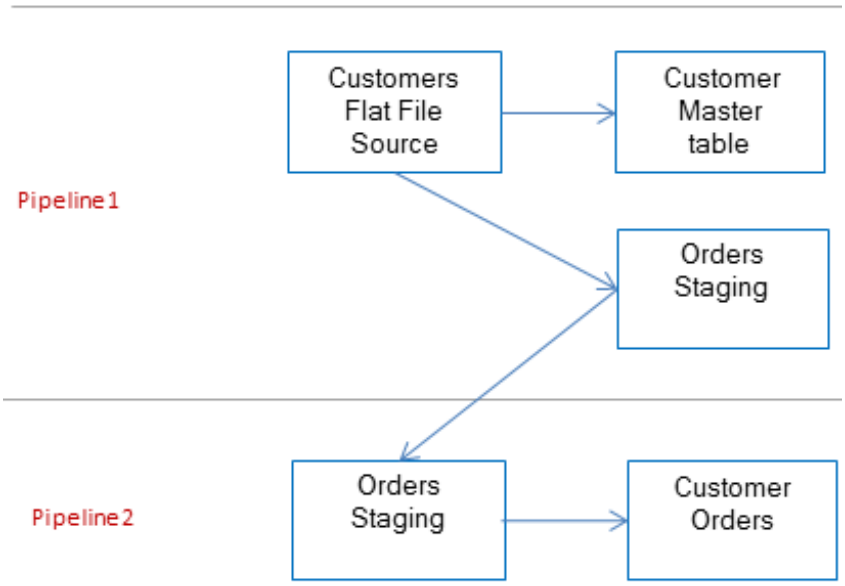
The following image shows the mapping:



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 Configurations

A mapping configuration controls the mapping deployment properties that the Developer tool uses when you run a mapping.

You can configure a default mapping configuration, a reusable mapping configuration, or a custom mapping configuration. The default mapping configuration specifies the default mapping properties. The Data Integration Service uses the default mapping configuration to run a mapping unless a reusable or custom mapping configuration is specified.

The following table describes the mapping configuration types:

Mapping Configuration	Description
Default mapping configuration	The default mapping configuration defines the default mapping deployment properties. The Data Integration Service runs a mapping using the default mapping configuration unless a reusable or custom mapping configuration is specified. Configure a default mapping configuration in the Developer tool preferences.
Reusable mapping configuration	Use a reusable mapping configuration to override the default mapping configuration when you run a mapping. The mapping configuration persists for the current mapping run. Configure a reusable mapping configuration in the **Run Configurations** dialog box. Select the mapping configuration and run the mapping through the **Run Configurations** dialog box, or select the mapping configuration when you run the mapping using advanced options. You must select the mapping configuration each time that you run a mapping.
Custom mapping configuration	Use a custom mapping configuration to override the default mapping configuration when you run a mapping. The custom mapping configuration properties include only the Data Integration Service, operating system profile for the Data Integration Service, override tracing level, and optimizer level. The custom mapping configuration persists for the current mapping run. Configure a custom mapping configuration in the **Run Mapping Using Advanced Options** dialog box. You must reconfigure the custom mapping configuration each time that you run a mapping.

Mapping Configuration Properties

Configure a mapping configuration in the Run Configurations dialog box, or configure a default mapping configuration in the Developer tool preferences. The properties that you can configure for the mapping include Data Integration Service properties, source properties, and advanced properties.

Data Integration Service Propeties

The following table displays the properties that you configure for the Data Integration Service:

Property	Description
Use the default Data Integration Service	Uses the default Data Integration Service to run the mapping. Default is enabled.
Data Integration Service	Specifies the Data Integration Service that runs the mapping if you do not use the default Data Integration Service.
Available operating system profiles	Specifies the operating system profile to run the mapping when the Data Integration Service is enabled to use operating system profiles. The Developer tool displays this property only if the administrator assigned at least one operating system profile to the user. The Data Integration Service runs the mapping with the default operating system profile assigned to the user. You can change the operating system profile from the list of available operating system profiles.

Source Properties

The following table displays the properties that you configure for sources:

Property	Description
Read all rows	Reads all rows from the source. Default is enabled.
Read up to how many rows	Specifies the maximum number of rows to read from the source if you do not read all rows. **Note:** If you enable the this option for a mapping that writes to a customized data object, the Data Integration Service does not truncate the target table before it writes to the target. Default is 1000.
Read all characters	Reads all characters in a column. Default is disabled.
Read up to how many characters	Specifies the maximum number of characters to read in each column if you do not read all characters. The Data Integration Service ignores this property for SAP sources. Default is 4000.

Advanced Properties

The following table displays the advanced properties:

Property	Description
Default date time format	Date/time format the Data Integration Services uses when the mapping converts strings to dates. Default is MM/DD/YYYY HH24:MI:SS.
Override tracing level	Overrides the tracing level for each transformation in the mapping. The tracing level determines the amount of information that the Data Integration Service sends to the mapping log files. Choose one of the following tracing levels: - None. The Data Integration Service uses the tracing levels set in the mapping. - Terse. The Data Integration Service logs initialization information, error messages, and notification of rejected data. - Normal. The Data Integration Service logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes mapping results, but not at the level of individual rows. - Verbose initialization. In addition to normal tracing, the Data Integration Service logs additional initialization details, names of index and data files used, and detailed transformation statistics. - Verbose data. In addition to verbose initialization tracing, the Data Integration Service logs each row that passes into the mapping. Also notes where the Data Integration Service truncates string data to fit the precision of a column and provides detailed transformation statistics. Default is None.
Sort order	Order in which the Data Integration Service sorts character data in the mapping. Default is Binary.
Optimizer level	Controls the optimization methods that the Data Integration Service applies to a mapping as follows: **Auto** The Data Integration Service applies optimizations based on the execution mode and mapping contents. **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. **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. Default is Auto.

Property	Description
High precision	Runs the mapping with high precision. High precision data values have greater accuracy. Enable high precision if the mapping produces large numeric values, for example, values with precision of more than 15 digits, and you require accurate values. Enabling high precision prevents precision loss in large numeric values. Default is enabled.
Send log to client	Allows you to view log files in the Developer tool. If you disable this option, you must view log files through the Administrator tool. Default is enabled.

How to Configure the Default Mapping Configuration

Update the default mapping configuration properties in the Developer tool preferences. The Developer tool uses the default mapping configuration properties to run mappings if no mapping configuration is specified.

1. In the Developer tool, click **Windows > Preferences**.

   The **Preferences** dialog box appears.
2. Navigate to **Informatica > Run Configurations > Mapping**.
3. Configure the default mapping configuration properties.
4. Click **OK**.

   The Developer tool updates the default mapping configuration properties.

How to Create a Reusable Mapping Configuration

Create a reusable mapping configuration in the Run Configurations dialog box. Specify the mapping configuration properties. Select the mapping configuration when you run a mapping in the Developer tool.

1. In the Developer tool, click **Run > Open Run Dialog**.

   The **Run Configurations** dialog box appears.
2. Right-click **Mapping Configuration** and select **New**.
3. Select the new mapping configuration.
4. Enter the mapping configuration name.
5. Configure the mapping configuration properties.

How to Create and Use a Custom Mapping Configuration

Configure a custom mapping configuration before you run a mapping using advanced options. You must reconfigure the custom mapping configuration each time that you run the mapping.

1. In the Developer tool, right-click a mapping in the editor or the Object Explorer view. Select **Run Mapping Using Advanced Options**.

   The **Run Mapping Using Advanced Options** dialog box appears.
2. Select **Specify a custom mapping configuration**.

3. Configure the custom mapping configuration properties. Specify the Data Integration Service, operating system profile for the Data Integration Service, override tracing level, and optimizer level.
4. Click **Run**.

   The mapping runs using the custom mapping configuration.

Advanced Mapping Options

You can run a mapping in the Developer tool using advanced options. The advanced mapping options include mappings configuration options and mapping parameter options.

Specify the mapping configuration and mapping parameters each time that you run the mapping. Select a reusable mapping configuration or customize the mapping configuration that you use when you run the mapping. Use the default parameter values in the mapping or use parameter values in a parameter set or a parameter file. The mapping configuration and mapping parameters that you specify persist for the current mapping run.

If you use advanced options to run a mapping instance where the parameters are resolved, you cannot specify mapping parameters to run the mapping. The mapping runs using the parameters that are resolved in the mapping.

The following table describes the options that you use to specify a mapping configuration:

Option	Description
Select a mapping configuration	Select a mapping configuration from the drop-down menu. To create a new mapping configuration, select New Configuration.
Specify a custom mapping configuration	Create a custom mapping configuration that persists for the current mapping run.

The following table describes the properties that you configure when you specify a custom mapping configuration:

Property	Description
Use the default Data Integration Service	Uses the default Data Integration Service to run the mapping. Default is enabled.
Data Integration Service	Specifies the Data Integration Service that runs the mapping if you do not use the default Data Integration Service.
Available operating system profiles	Specifies the operating system profile to run the mapping when the Data Integration Service is enabled to use operating system profiles.
Override tracing level	Overrides the tracing level for each transformation in the mapping. The tracing level determines the amount of information that the Data Integration Service sends to the mapping log files.
Optimizer level	Controls the optimization methods that the Data Integration Service applies to a mapping.

The following table describes the options that you use to specify mapping parameters:

Mapping Parameters	Description
Apply the default values in the mapping	Resolves the mapping parameters based on the default values configured for the parameters in the mapping. If parameters are not configured for the mapping, no parameters are resolved in the mapping.
Apply a parameter set	Resolves the mapping parameters based on the parameter values defined in the specified parameter set.
Apply a parameter file	Resolves the mapping parameters based on the parameter values defined in the specified parameter file.

Note: If the mapping does not contain parameters, the option to specify mapping parameters is disabled.

How to Run a Mapping Using Advanced Options

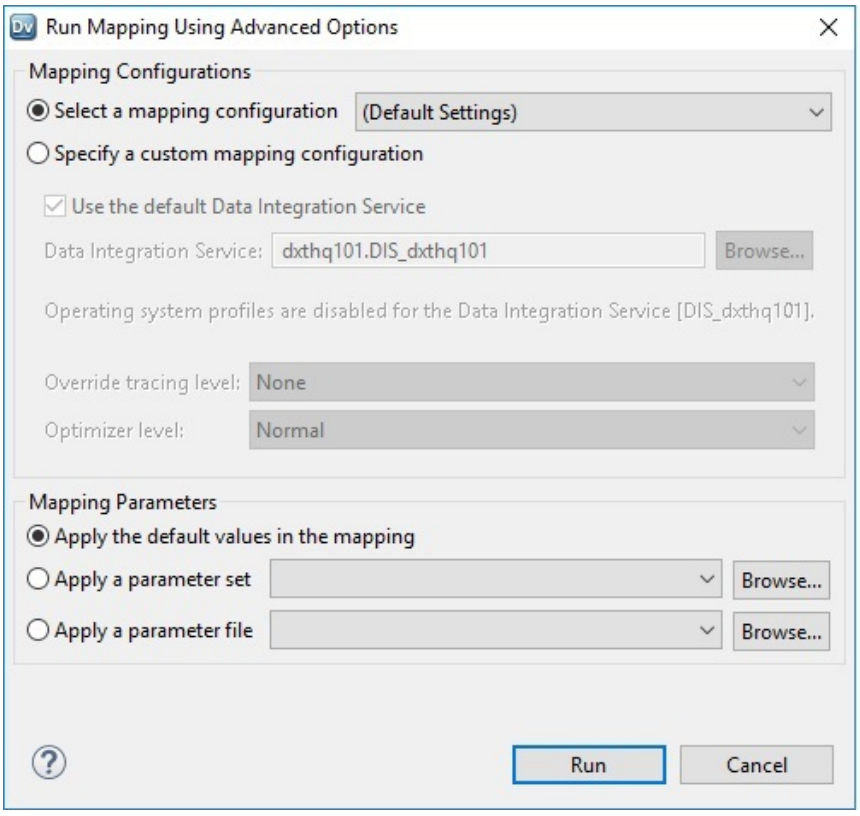
To run a mapping using advanced options, run the mapping through the **Run Mapping Using Advanced Options** dialog box. Select a mapping configuration or create a custom mapping configuration, and specify

the mapping parameters. The mapping configuration and mapping parameters that you specify persist for the current mapping run.

If you use advanced options to run a mapping instance where the parameters are resolved, you cannot specify mapping parameters to run the mapping. The mapping runs using the parameters that are resolved in the mapping.

1. In the Developer tool, right-click a mapping in the editor or the Object Explorer view. Select **Run Mapping Using Advanced Options**.

   The **Run Mapping Using Advanced Options** dialog box appears.



2. Select one of the following mapping configuration options:
   - Select a mapping configuration. Choose a reusable mapping configuration to run the mapping.
   - Specify a custom mapping configuration. Select the Data Integration Service, operating system profile, override tracing level, and optimizer level.
3. Optional. If the mapping contains parameters, you can select one of the following parameter options:
   - Apply the default values in the mapping. The Data Integration Service applies the default parameter values configured in the mapping.
   - Apply a parameter set. The Data Integration Service applies the parameter values configured in the parameter set.
   - Apply a parameter file. The Data Integration Service applies the parameter values configured in the parameter file.

How to Develop a Mapping

Develop a mapping to read, transform, and write data according to your business needs.

To develop and run a mapping, perform the following tasks:

1. Determine the type of mapping that you want to create.
2. Create input, output, and reusable objects such as the reusable transformations and mapplets that you want to use in the mapping.
   You can use physical data objects, logical data objects, or virtual tables for mapping input and output.
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.
8. Run the mapping to see the 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.

Adding Objects to a Mapping

Add objects to a mapping to determine the data flow between sources and targets.

- Drag a data object to the editor and select Read to add the data object as a source.
- Drag a data object to the editor and select Write to add the data object as a target.
- 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.
- 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 that you want to add.
- 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.
- Configure ports and properties for each non-reusable transformation.
- Optionally, drag a mapplet to the editor.

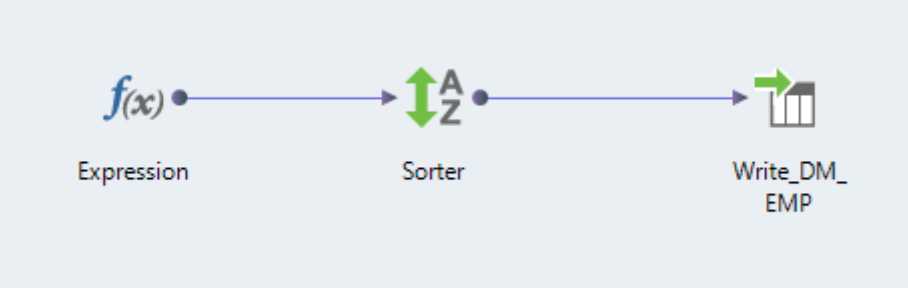
Connecting Mapping Objects

Complete a mapping by connecting the mapping objects. You connect mapping objects through the ports. Data passes into and out of a transformation through input ports, output ports, and input/output ports.

When you add an object to a mapping, you connect the properties according to how you want the Data Integration Service to transform the data. The editor displays mapping objects in the following ways:

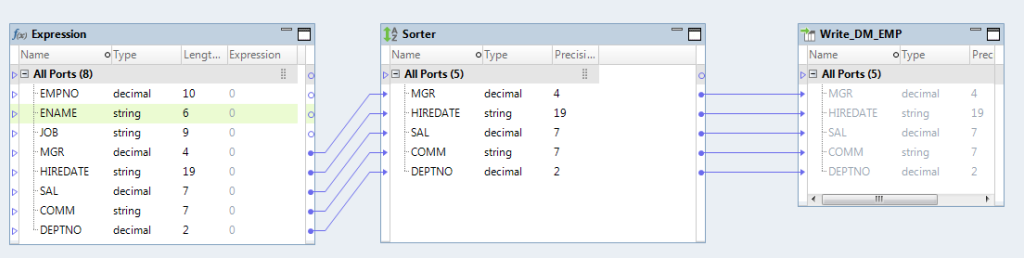
- Iconized. Shows an icon of the object with the object name.

  The following image shows a mapping with iconized objects:



- Normal. Shows the columns and the input and output port indicators. You can connect objects that are in the normal view.

  The following image shows the preceding iconized mapping in normal view:



When you link ports between input objects, transformations, mapplets, and output objects, you can create the following types of links:

- One to one links. Link one port in an input object to one port in an output object.
- One to many links. Link one port to multiple output objects. You can also link multiple ports in one object to multiple output objects.

You can manually link ports or link ports automatically:

- Manually linking ports. You can manually link one port or multiple ports. Drag a port from an input object to the port of an output object.
- Automatically linking ports. When you link ports automatically, you can link by position or by name.

For more information about linking ports, see the *Developer Transformation Guide*.

Creating Target Load Order Constraints

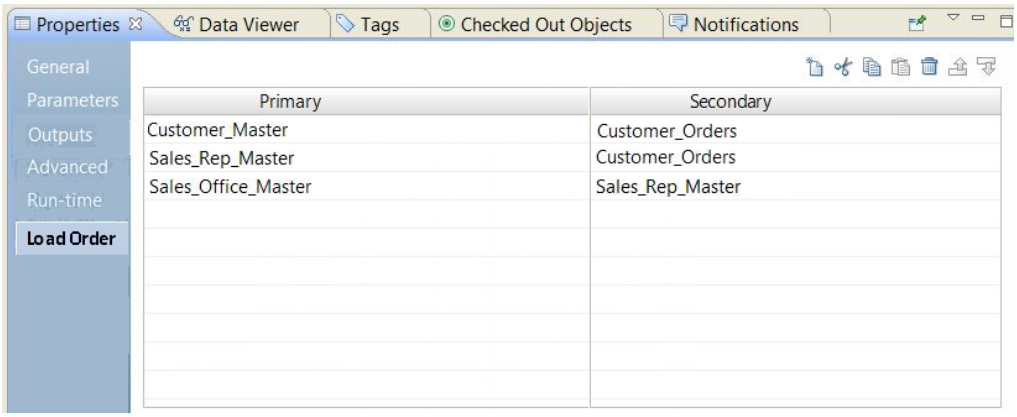
Target load order constraints restrict the order that the Data Integration Service loads rows to target tables in the same mapping. Create target load order constraints on the **Load Order** tab of a mapping.

1. Click inside the editor.

   The mapping **Properties** tabs appear in the bottom window.

2. Click the **Load Order** tab.

   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.

3. To enter a constraint, click the **New** button.

   The Developer tool creates a row for the constraint.

4. Click the **Primary** field.

   A list of the target instances in the mapping appears.

5. Select the target instance that you want to load first.

6. In the **Secondary** field, select the target instance to load second.

   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:

Primary	Secondary
Sales_Office_Master	Sales_Rep_Master
Sales_Rep_Master	Customer_Master
Customer_Master	Customer_Orders

7. Enter as many constraints as you need.

Validating a Mapping

Validate a mapping to ensure that the Data Integration Service can read and process the entire mapping.

1. Right-click the mapping and select **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 the domain includes more than one Data Integration Service and you have not selected a default service, the Developer tool prompts you to select one when you preview data or run a mapping.

- Right-click the mapping in the editor or the Object Explorer view and click **Run Mapping**.

  The Data Integration Service runs the mapping and writes the output to the target.
  When the Data Integration Service is configured to use operating system profiles, it runs the mapping with the operating system profile.

CHAPTER 2

Mapplets

This chapter includes the following topics:

- Mapplets Overview, 42
- Mapplet Types, 42
- Mapplet Input and Output, 43
- Generated Mapplets, 44
- Rule Specifications and Mapplets, 46
- Creating a Mapplet, 47
- Mapplet Validation, 47

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.

You can create a mapplet manually. You can also generate a mapplet from a segment within a mapping or mapplet.

Mapplet Types

The mapplet type is determined by the mapplet input and output.

You can create or generate 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.

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.

When you use an Input transformation to provide the input for a mapplet, you can set default values in the output ports of the Input transformation. The default values in the Input transformation output ports become default values for the mapplet input ports. If NULL data is passed from the mapping to the Input transformation, the Data Integration Service replaces the NULL value with the default value. The mapplet input uses the default value.

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.

Input transformations can receive data from a single active source. Unconnected ports do not appear in the mapping editor.

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.

Generated Mapplets

You can generate a mapplet from a segment in a mapping or mapplet. You might want to generate a mapplet when a mapping or mapplet contains a connected transformation flow that you want to reuse.

The Developer tool validates the segment as a mapplet as part of the generation process. Review the rules and guidelines for generated mapplets to avoid validation errors.

Generated Mapplet Rules and Guidelines

Mapplet generation fails if any of the following conditions are true:

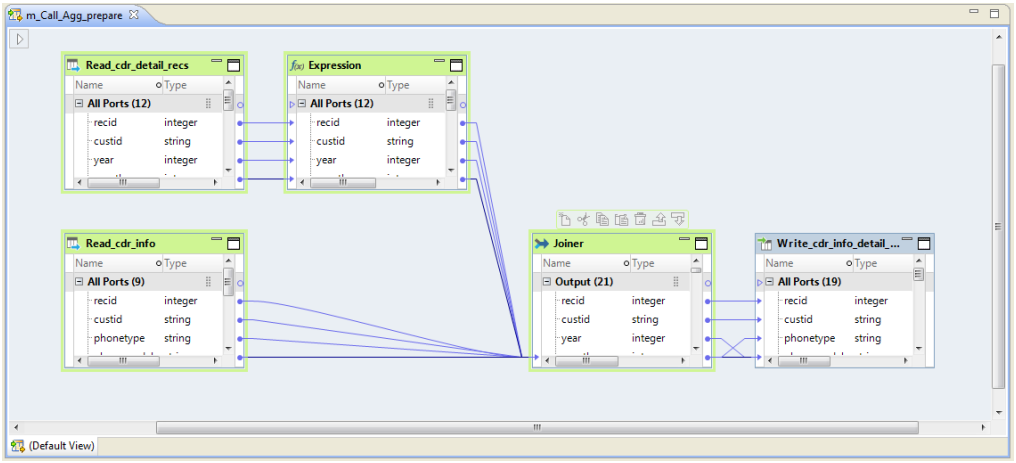
- The selected transformations are not in sequence.
- The segment contains both Read and Write transformations. However, a segment can include multiple Read or multiple Write transformations.
- The segment includes nonreusable Sequence Generator transformations, Input transformations, Output transformations, or transformations containing set operations.
- The selected segment does not include all transformations in a pipeline branch.
- The first and last transformations in a segment contain dynamic fields.
- The segment includes incoming run-time links to the first transformation or outgoing run-time links from the last transformation.
- The segment consists of a single parameterized Read, Write, or Lookup transformation.

Generating a Mapplet

Generate a mapplet from a segment containing connected transformations. The segment can contain Read, Write, or midstream transformations.

1. Open the mapping or mapplet that contains the segment you want to generate into a mapplet.
2. Select the transformations to include in the mapplet.
3. Right-click one of the selected transformations, and select **Extract Mapplet**.

   The following image shows a mapping with four transformations selected:



The generation process validates the segment and reports any validation errors.

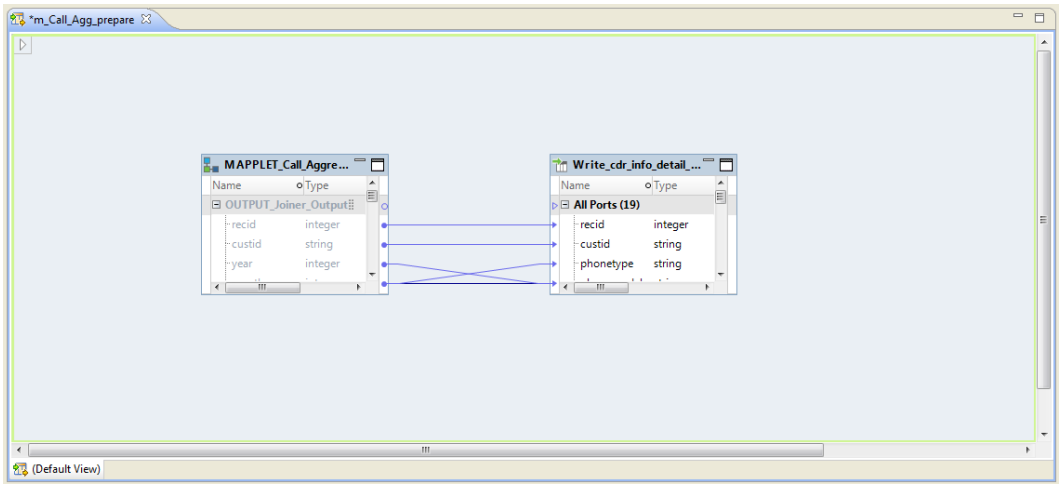
4. Browse to the Mapplets node you want to generate the mapplet in.

   The Developer tool generates the mapplet in the Mapplets node within the current project by default.

5. Click **Finish**.

   The mapplet replaces the transformations selected in the original mapping or mapplet. The tool adds Input or Output transformations to the mapplet based on whether the segment contains Read, Write, or midstream transformations.

The following image shows the selected transformations replaced by the mapplet:



Note that you must explicitly save the modified mapping or mapplet to replace the selected transformations with the mapplet. To return the mapping or mapplet to its original state, select **File > Undo** three times.

Rule Specifications and Mapplets

A rule specification is a Model repository object that uses business logic to describe transformation operations. Users create rule specifications in Informatica Analyst. You can add a rule specification to a mapping in the same way that you add a mapplet to a mapping.

You can also add a rule specification to a mapplet and deploy a rule specification from the Developer tool as a web service.

An Analyst tool user can generate one or more mapplets from a rule specification. Each mapplet contains transformations that represent the rule specification logic. When you run a mapping that contains either the rule specification or the corresponding mapplet, you get the same results.

You can edit a mapplet that a user generates from a rule specification in the same way as any mapplet that you create in the Developer tool. You cannot edit a rule specification in the Developer tool. Add a rule specification to a mapping when you want the mapping to apply the logic that the rule specification represents. Add the corresponding mapplet to a mapping when you want to use or update the mapplet logic independently of the rule specification.

Rules and Guidelines for Rule Specifications

- A rule specification contains a primary rule set and optionally contains additional rule sets. The primary rule set represents the complete logic of the rule specification. Additional rule sets define discrete data analysis operations and provide outputs that other rule sets can read.

  The mapplet that represents the primary rule set has the same name as the rule specification.
- If you rename a rule specification in the Developer tool, the Analyst tool displays the name when the user opens the rule specification. If you rename the mapplet for the primary rule set, you do not change the rule specification name.
- If an Analyst tool user adds, deletes, or edits an input in a rule specification, the user breaks all input links to other objects in a mapping. If an Analyst tool user adds, deletes, or edits an output in a rule specification, the user breaks all input links to other objects in a mapping. The edits that break the links include changes to the name, precision, or data type of an input or output. Update the links in any mapping that uses the rule specification.

  If an Analyst tool user updates the business logic in a rule specification but does not alter the inputs or outputs, the input and output links do not break. The changes that the user makes are available to the mapping when the user saves the rule specification.

Rule Specification Properties

A rule specification has properties that you can view and edit in the Developer tool. Use the properties to review descriptive metadata for the rule specification name. You can also use the properties to determine the number of output ports that the rule specification generates for downstream mapping objects.

To view the properties, open a mapping that contains the rule specification and select the rule specification icon. Then, select the **Properties** tab in the mapping.

The tab displays the following views:

General

The general properties contain the name and description of the rule specification instance.

If you update the rule specification name or description in a mapping, the changes that you make apply in the current mapping only.

Properties

The properties include the rule specification name that appears in the General view. The properties also specify any date range that an Analyst tool user set for the rule specification. The date range indicates the interval during which the rule specification is valid for use.

Note: You can select or clear an option to allow an output for each child rule in the rule specification. A child rule is a rule set in the rule specification. When you select the option, the Developer tool adds an output port for each rule set to the rule specification in the mapping. The option is clear by default. Select the option to make available the rule set outputs to downstream objects in the mapping.

Ports

The port properties list the input and output ports on the rule specification instance. The port properties display the name, data type, precision, and scale of each port. You can optionally add a description to a port. The description applies to the port in the current rule specification instance.

Run-time Linking

The run-time linking properties determine now the rule specification ports link to other objects in a dynamic mapping.

Advanced

The advanced properties include the tracing level setting. The tracing level defines the amount of detail that appears in the log for the rule specification. You can choose terse, normal, verbose initialization, or verbose data. The default value is normal.

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.

Mapplet Validation

You can validate a mapplet before you add it to a mapping. You can also validate a mapplet to use as a rule in a profile.

Validating a Mapplet

Validate a mapplet before you add the mapplet to a mapping. You can also validate a mapplet as a rule to include in a profile.

1. Right-click the mapplet editor.
2. Select **Validate As** > **Mapplet** or **Validate As** > **Rule**.

   The Validation Log displays any errors that occur.

Mapplet as a Rule Validation

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. You can validate a mapplet that you want to use as a rule in a profile.

A rule must meet the following requirements:

- The rule must contain an Input and Output transformation. You cannot use data sources in a rule.
- The rule can contain Expression transformations, Lookup transformations, and passive data quality transformations. The rule cannot contain any other type of transformation. For example, a rule cannot contain a Match transformation, as it is an active transformation.
- The rule 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.

CHAPTER 3

Mapping Parameters

This chapter includes the following topics:

- Mapping Parameters Overview, 49
- System Parameters, 50
- User-Defined Parameters, 51
- Where to Create User-Defined Parameters, 53
- Parameters in Mappings, 53
- Parameters in Mapplets, 55
- Parameters in Logical Data Objects, 57
- Parameters in Virtual Table Mappings, 58
- Parameter Sets , 59
- Parameter Files, 60
- Parameter Hierarchy, 65
- How to Configure Parameters, 68
- How to Run a Mapping with Parameters, 78

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 127

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 Analyst tool displays the file path of system parameters in the following format: `$$[Parameter Name]/[Path]`. For example, `"$$SourceDir/ff_dept.txt."`

The following table describes the system parameters:

System Parameter	Type	Description
CacheDir	String	Default directory for index and data cache files.
LogDir	String	Default directory for Mapping task log files.
RejectDir	String	Default directory for reject files.
SourceDir	String	Default directory for source files.
TargetDir	String	Default directory for target files.
TempDir	String	Default directory for temporary files.
ApplicationName	String	Name of the application

System Parameter	Type	Description
ExecutionEnvironment	String	Hadoop or Native environment.
MappingName	String	Name of the mapping that is running.
MappingRunStartTime	Date/time	The start time of the mapping that is running.
ServiceName	String	The Data Integration Service name.
UserName	String	Name of the user that is running the mapping.

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 single port. You can use the port parameter in the Rank port of the Rank transformation.

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.

Sort Key List

A list of ports to sort with order keys in an Expression transformation configured for windowing. This 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 create a parameter, you cannot include a dollar sign ($) as the leading character in the parameter name.

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.

A parameter cannot contain a series of values. If you provide a series of values in a parameter, the Data Integration Service treats the parameter values as a single string value.

For example, you have the parameters $IndexParameter1 (value 2) and $IndexParameter2 (value1, value2, value3). You include these parameters in the expression INDEXOF as:

```
INDEXOF($IndexParameter1,'value1','value2','value3')
```

The Data Integration Service returns the value 0 instead of the value 2.

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 can create a parameter in a Read transformation to parameterize the columns that the transformation reads from a data source. Or, you can create a mapping parameter that defines the run-time environment.

You can create parameters for the following objects:

- Transformations or data objects
- Logical data objects
- Mapplets
- Mappings
- Workflows

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.

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

Maintain user-defined parameters on 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.

You can also access parameters directly from the Outline view, which shows where parameters are used, defined, and bound. When you click on a parameter in the Outline view, the parameter properties appear on the **Parameters** tab.

Parameters in Mappings

You can assign parameters to mapping properties, or you can create parameters for mapping objects.

When you define a mapping parameter, you can bind the mapping parameter to a 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 reusable transformation, reusable mapplet, or reusable data source parameter values with a parameter set or parameter file. To change parameter values in reusable objects at run time, expose the parameter values as mapping parameters. Configure mapping parameters for the exposed values in the parameter set or parameter file.

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. You can also view and access the parameter in the Outline view.

Add mapping parameters from reusable transformation parameters

After you add a transformation to a mapping, you can create a mapping parameter directly from the transformation **Parameters** tab. To create a mapping parameter from the transformation parameter, expose the transformation parameter as a mapping parameter. The Developer tool creates a mapping parameter that has the same name and type as the transformation parameter.

Add parameters to a non-reusable transformation

If you create a transformation in a mapping, the transformation is a non-reusable transformation. If you parameterize any of the transformation properties, you create mapping parameters instead of transformation parameters. The mapping parameters bind to the transformation parameters.

Bind mapping parameters from mapping outputs

If you define a mapping output, you can bind it to an input parameter in the same mapping. Define the mapping output on the **Outputs** tab. On the **Parameters** tab, define a mapping parameter and use the **Binding** column to bind the output you defined earlier.

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 and bind the mapping parameter to the transformation parameter in the same step. If you click the **Expose as Mapping Parameter** button a second time, and the transformation parameter is already bound to mapping parameter, the Developer tool does not change the mapping parameter.

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. After you define a parameter, you can view and access the parameter in the Outline view.

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 a different name than the transformation parameter and you can set a different default value.

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.

Use default

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

Value

Enter a default parameter value to use in the mapping. Skip binding a mapping parameter to 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:

Object Name	Object Type	Parameter Name	Parameter Default Value
mp_Get_Customer	Mapplet	mp_cust_connection	Oracle_Default
m_billing_mapping	Mapping	m_acctg_connection	Oracle_AcctDB

Object Name	Object Type	Parameter Name	Parameter Default Value
m_order_fill_mapping	Mapping	m_shipping_connection	Oracle_Warehouse
m_cust_main_mapping	Mapping	m_master_connection	Oracle_Cust_Mast

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

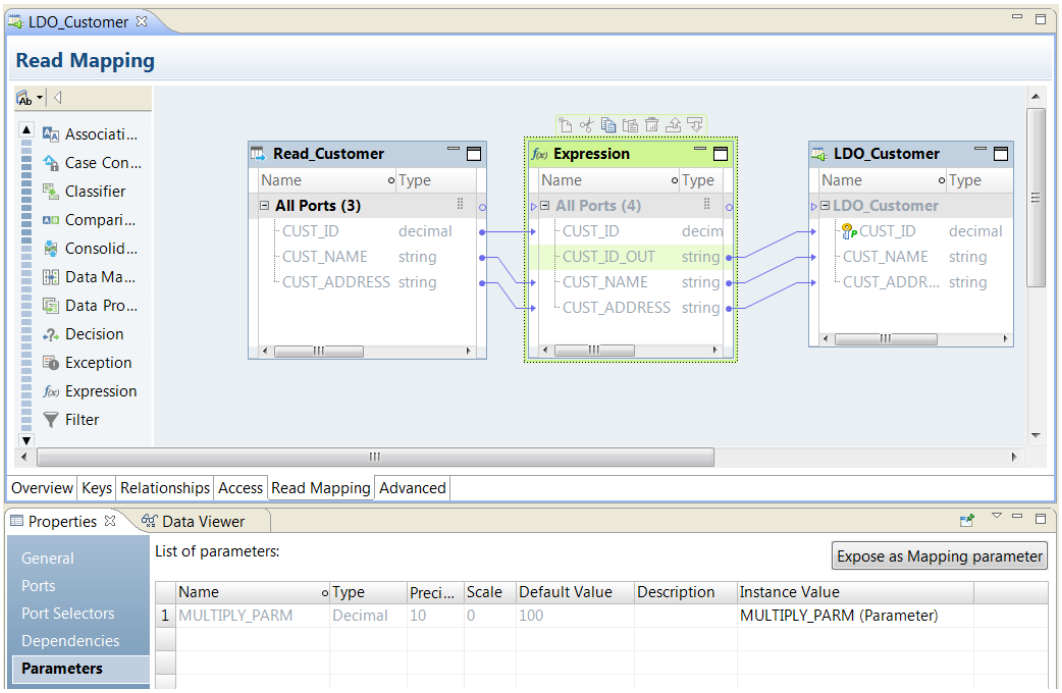
You can include parameters in logical data objects. You can use them in transformations and in the Read and Write mappings.

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:



To view the duplicate parameter, select the logical data object in the Outline view. 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

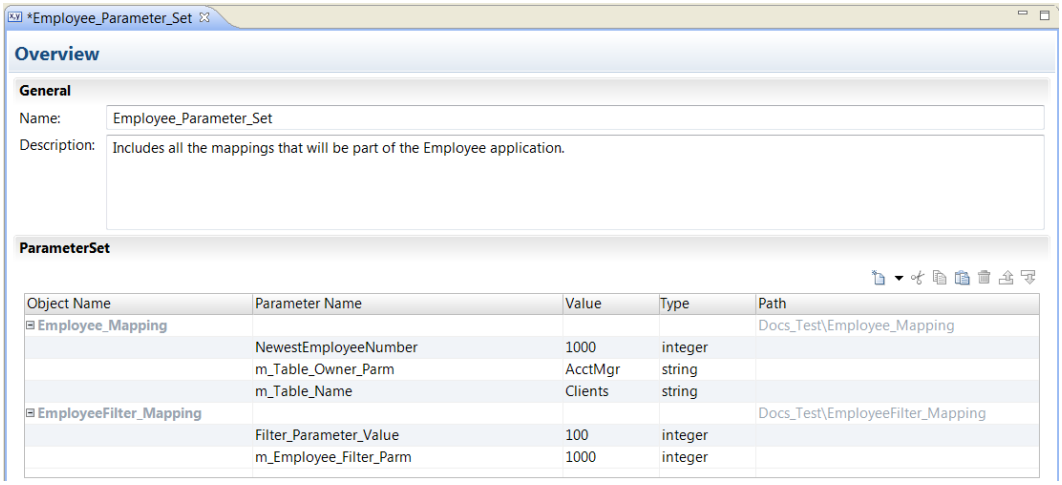
A parameter set is an object in the Model repository that contains a set of parameters and parameter values to run mappings and workflows.

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.

Note: The parameter type that you specify in the parameter set must match the parameter type in the mapping, Mapping task, or workflow. If the parameter types do not match, the mapping, Mapping task, or workflow uses the default value for the parameter.

When you use a parameter set to configure mapping parameter values, the link between the parameter set and the mapping depends on the project where the mapping is defined. If the project name is changed, you must re-establish the link.

To re-establish the link, edit the parameter set and re-select the mapping that uses the parameter set.

For example, you have a project named P1 that includes a mapping m_1 that uses a parameter set ps_1. You change the project name to P2. Subsequently, you must edit the parameter set to re-select mapping m_1.

Parameter Files

A parameter file is an XML file that lists user-defined parameters and their assigned values. Parameter files provide the flexibility to change parameter values each time you run a mapping.

The parameter values define properties for a mapping. The Data Integration Service applies these values when you run a mapping and specify a parameter file.

You can define mapping parameters and workflow parameters in a parameter file. If you want to specify reusable object parameters, expose the reusable object parameters as mapping parameters. Specify the mapping parameter values in the 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.

You can export a parameter file from the Developer tool. Export the file from the mapping or the workflow **Parameters** tab. The Developer tool generates a parameter file that contains the mapping or workflow parameters and the default parameter values. You can specify the parameter file name and choose where to save the 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.

You can also list the parameters and default values used in a mapping from the command line. You can use the command line output as a parameter file template.

Run a mapping with a parameter file from the Developer tool or run the mapping from the command line.

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 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. 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 mapping in a specific deployed application. If you define the same parameter in a project top-level element and an application top-level element in the same parameter file, the parameter value defined in the application element takes precedence.

The Data Integration Service searches for parameter values in the following order:

1. The value specified within an application element.
2. The value specified within a project element.
3. The parameter default value.

A parameter file must conform to the structure of the parameter file XML schema definition (XSD). If the parameter file does not conform to the schema definition, the Data Integration Service fails the mapping run.

On the machine that hosts the Developer tool, the parameter file XML schema definition appears in the following directory:

```
<Informatica Installation Directory>\clients\DeveloperClient\infacmd\plugins\ms
\parameter_file_schema_1_0.xsd
```

On the machine that hosts Informatica Services, the parameter file XML schema definition appears in the following directory:

```
<Informatica Installation Directory>\isp\bin\plugins\ms\parameter_file_schema_1_0.xsd
```

Project Element

A project element defines the parameter values to use when you run a specific mapping in the project. 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 transformation, mapplet, or data object elements in the project element.

The following table describes the elements that a project element can contain:

Element Name	Description
folder	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 mapping or a workflow element, or another folder element.
mapping	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 transformation, mapplet, or data object in the mapping that accepts parameters.

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 within the project.

For example, you want the Data Integration Service to apply the parameter defined by "MyMapping_Param" when you run the mapping "MyMapping". You can define the parameter in the parameter file using the following elements:

```
<project name="MyProject">

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

</project>
```

Application Element

An application element provides a run-time scope for a project element. The application element defines the deployed application and the parameter values that the Data Integration Service applies to a mapping or workflow in the deployed application.

You can use application elements in a parameter file to define different sets of parameters that the Data Integration Service applies to a mapping or workflow depending on which application runs.

The application element can contain project, folder, mapping or workflow elements. The following table describes the elements that the application element can contain:

Element Name	Description
project	Defines a project in the Model repository. Use a project element to locate a folder, mapping, or workflow that resides in the project.
folder	Defines a folder within a project. Use a folder element if objects are organized in multiple folders within the project. A folder element can contain a mapping or a workflow element, or another folder element.
mapping	Defines a mapping in the application that uses parameters. A mapping element contains one or more parameter elements that define parameter values for the mapping or for any non-reusable transformation, mapplet, or data object in the mapping that accepts parameters. You can specify the same mapping in different application elements to use different parameters depending on which application runs.

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:

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

- You can reference mapping and workflow parameters in a parameter file. You cannot reference reusable transformation, mapplet, or data object parameters. To reference reusable object parameters, expose the reusable object parameters as mapping parameters. Specify the mapping parameter values in the parameter file.
- The application element contains mapping or workflow parameters that apply only to the specified application when the application runs. If you run a mapping with a parameter file from the Developer tool and do not deploy the mapping as an application, do not specify an application element. Specify the mapping within a project element.

- Parameter values cannot be empty. For example, the Data Integration Service fails the mapping run if the parameter file contains the following entry:

  ```
  <parameter name="Param1"> </parameter>
  ```

- 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 version="1.0"?>
<root description="Sample Parameter File"
      xmlns="http://www.informatica.com/Parameterization/1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!--
        The Data Integration Service uses this section only when you run mapping "Map1"
or "Map2"
        in project "Project1" in deployed application "App1."

        This section assigns values to parameters created in mappings "Map1" and "Map2."
    -->
    <application name="App1">
        <mapping name="Map1">
            <project name="Project1">
                <mapping name="Map1">
                    <parameter name="MAP1_PARAM1">MAP1_PARAM1_VAL</parameter>
                    <parameter name="MAP1_PARAM2">MAP1_PARAM2_VAL</parameter>
                </mapping>
            </project>
        </mapping>
        <mapping name="Map2">
            <project name="Project1">
                <mapping name="Map2">
                    <parameter name="MAP2_PARAM1">MAP2_PARAM1_VAL</parameter>
                    <parameter name="MAP2_PARAM2">MAP2_PARAM2_VAL</parameter>
                </mapping>
            </project>
        </mapping>
    </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">
        <mapping name="Map1">
            <project name="Project1">
                <mapping name="Map1">
                    <parameter name="MAP1_PARAM2">MAP1_PARAM2_VAL</parameter>
                </mapping>
            </project>
        </mapping>
    </application>
</root>
```

Export a Parameter File

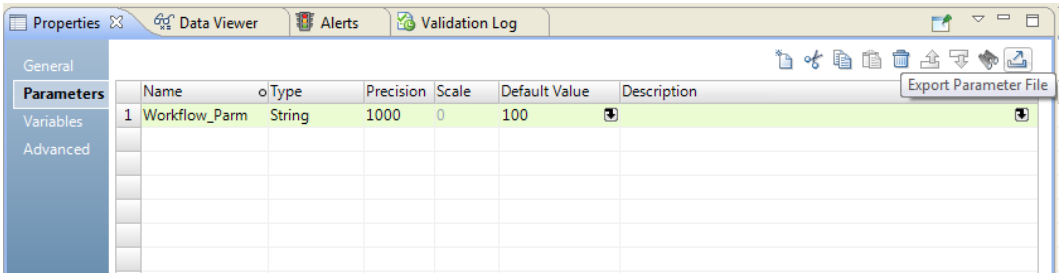
You can export a mapping parameter file or a workflow parameter file from the Developer tool. Define the parameters in the Developer tool and then export them to a file. The Developer tool creates a parameter file in .XML format.

You can export a parameter file that contains mapping parameters or workflow parameters. You can export parameters from the mapping **Parameters** tab or from the workflow **Parameters** tab. The Developer tool exports all the parameters from the **Parameters** tab.

To export a parameter file, perform the following steps:

1. Define the parameters and the parameter defaults for a mapping or a workflow.
2. On the **Parameters** tab of the mapping or workflow **Properties**, click the **Export Parameter File** option.
3. Enter a name for the parameter file and browse for a location to put the file.
4. Click **Save**.

The following image shows the **Export Parameter File** option on the Parameters tab for a workflow:



When you export a parameter file, the Developer tool creates a parameter file with either mapping parameters or workflow parameters in it. The Developer tool does not export mapping and workflow parameters to the same file.

For example, when you export the workflow parameter, Workflow_Parm, the Developer tool creates the following parameter file:

```
<?xml version="1.0" encoding="UTF-8"?>
-<root version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema" xmlns="http://
www.informatica.com/Parameterization/1.0">
   -<project name="Orders">
      -<workflow name="Customer_Workflow">
         <parameter name="Workflow_Parm">100</parameter>
      </workflow>
   </project>
</root>
```

Creating a Parameter File from infacmd ms ListMappingParams

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.

Parameter Hierarchy

The parameter hierarchy defines the levels where you can create user-defined parameters.

The hierarchy occurs in the following order:

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

A parameter assigned to an object at a higher level in the hierarchy can replace parameters assigned to lower levels in the hierarchy. You can use the hierarchy to override parameters in the following ways:

- Override parameters assigned to low-level objects in the hierarchy by assigning parameters to high-level objects in the hierarchy.
- Bind low-level object parameters to high-level object parameters to use the default values of the high-level object parameters in the run-time mapping or workflow.

For example, you can add a reusable transformation to a mapping. To replace parameters in the reusable transformation, you can create mapping parameters to override the transformation parameters. Or, you can expose the transformation parameters as mapping parameters to bind the transformation parameters to mapping parameters. The transformation uses the default mapping parameter values at run time.

Overriding Parameters Using the Parameter Hierarchy

When you create a parameter for a transformation, a logical data object, a mapplet, or a mapping, you can override the parameter by specifying a new parameter for the transformation, logical data object, mapplet, or mapping at a higher level in the parameter hierarchy.

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.

You can also override parameters using a parameter set or a parameter file. The parameters in the mapping use the parameters specified in the parameter set or parameter file. In a parameter set or a parameter file, you can specify only workflow and mapping parameters.

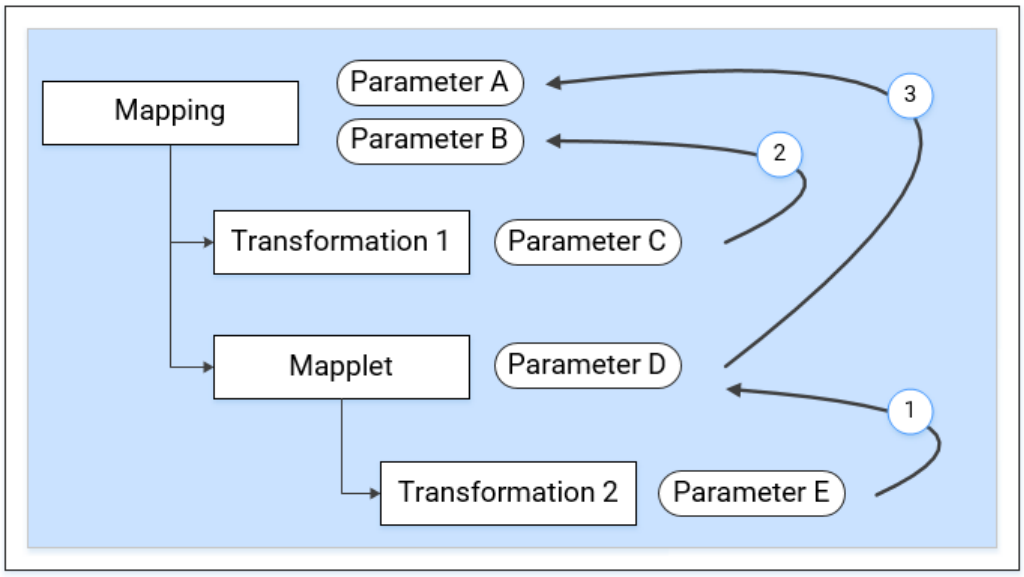
Binding Parameters to Override the Parameters at Run Time

When you add a reusable object to a mapping or workflow, bind parameters to higher-level parameters in the parameter hierarchy to override the default value of the reusable object parameter. At run time, the Data Integration Service applies the higher-level parameter to the reusable object.

For example, you create a reusable transformation and a parameter in the transformation. Then you add the reusable transformation to a mapplet. You can either use the default parameter value from the transformation, or you can bind the transformation parameter to the mapplet parameter. Change the default value for the mapplet parameter to override the default value of the transformation parameter at run time.

To bind user-defined parameters, expose the parameters. For example, if you add a reusable transformation to a mapping, expose the transformation parameters as mapping parameters to bind the transformation parameters to the mapping parameters.

The following image demonstrates how the Data Integration Service binds parameters in Developer tool mappings:



1. The parameter in the transformation binds to the parameter in the mapplet. The transformation parameter uses the default value of the mapplet parameter at run time.
2. The parameter in the mapplet binds to the parameter in the mapping. The mapplet parameter uses the default value of the mapping parameter at run time.
3. The parameter in the transformation binds to the parameter in the mapping. The transformation parameter uses the default value of the mapping parameter at run time.

Use Cases for Overriding Parameters in Mappings

There are different guidelines to override parameters in non-reusable and reusable mapping objects. You can override both non-reusable and reusable object parameters by creating or reconfiguring mapping parameters. For reusable object parameters, you can first expose the object parameters as mapping parameters. Then reconfigure the mapping parameters.

To override mapping parameters, specify a parameter set or parameter file when you run the mapping.

Overriding Non-Reusable Transformation Parameters in Mappings

When you create a non-reusable transformation in a mapping, the transformation parameters are mapping parameters.

To override the default parameter values in the transformation, complete one of the following tasks:

- Reconfigure the mapping parameters.
- Configure a parameter set or parameter file to override the mapping parameters.

For example, you can create a non-reusable Filter transformation in a mapping and set a parameter for the Filter transformation. The Developer tool duplicates the transformation parameter and creates a mapping parameter. To override the transformation parameter, reconfigure the default value of the mapping parameter, or configure a parameter set or parameter file to override the mapping parameter.

Overriding Reusable Transformation Parameters in Mappings

When you add a reusable transformation to a mapping, the transformation parameters are transformation parameters until the parameters are exposed as mapping parameters. If you expose the transformation parameters as mapping parameters, you bind the transformation parameters to the mapping parameters. The mapping parameters have the same name and type as the transformation parameters. The default value of the mapping parameter uses the instance value of the reusable transformation parameter.

To override the parameter value in a reusable transformation, complete one of the following tasks:

- Create a new mapping parameter to override the transformation parameter.
- Expose the transformation parameters as mapping parameters. Reconfigure the default value of the mapping parameters.
- Expose the transformation parameters as mapping parameters. Configure a parameter set or a parameter file to override the default value of the mapping parameters.

For example, you can set a parameter for a reusable Filter transformation. When you add the reusable Filter transformation to a mapping, you can configure a mapping parameter to override the Filter transformation parameter, or you can expose the Filter transformation parameter as a mapping parameter. If you expose the Filter transformation parameter as a mapping parameter, the Filter transformation parameter binds to a mapping parameter. To override the exposed transformation parameter, reconfigure the default value of the mapping parameter that binds to the transformation. The transformation uses the default value of the mapping parameter at run time.

Overriding Non-Reusable Transformation Parameters in a Reusable Mapplet

When you create a reusable mapplet that contains a non-reusable transformation, the transformation remains non-reusable. If you create a non-reusable transformation in a mapplet, the transformation parameters are mapplet parameters. When you add a reusable mapplet that contains a non-reusable transformation to a mapping, the non-reusable transformation in the mapplet remains a non-reusable transformation in the mapping.

You can expose the mapplet parameters as mapping parameters. When you expose the mapplet parameters, the mapplet parameters bind to mapping parameters, and the non-reusable transformation parameters become mapping parameters.

To override the reusable mapplet parameters, complete one of the following tasks:

- Configure mapping parameters.

- Expose the mapplet parameters as mapping parameters. After you expose the mapplet parameters, the mapplet parameters use the default values of the mapping parameters at run time. Reconfigure the mapping parameters.
- Expose the mapplet parameters as mapping parameters. After you expose the mapplet parameters, the mapplet parameters use the default values of the mapping parameters at run time. Configure a parameter set or a parameter file to override the mapping parameters.

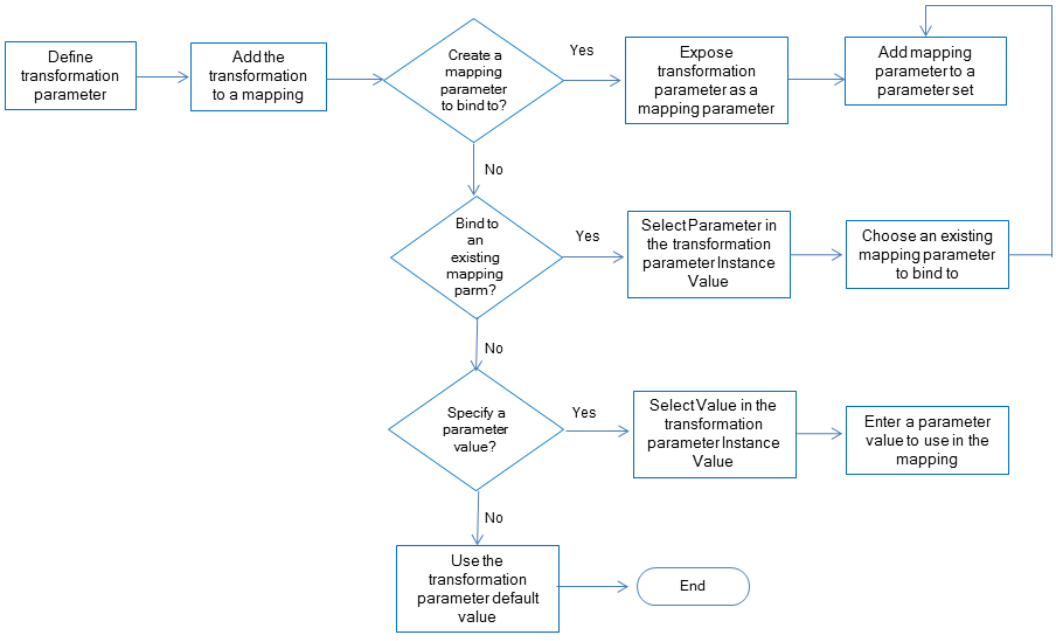
To override the non-reusable transformation parameters, complete one of the following tasks:

- Reconfigure the mapplet parameters.
- Expose the mapplet parameters as mapping parameters. After you expose the mapplet parameters, the non-reusable transformation parameters become mapping parameters. Reconfigure the mapping parameters.
- Expose the mapplet parameters as mapping parameters. After you expose the mapplet parameters, the non-reusable transformation parameters become mapping parameters. Configure a parameter set or a parameter file to override the mapping parameters.

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.

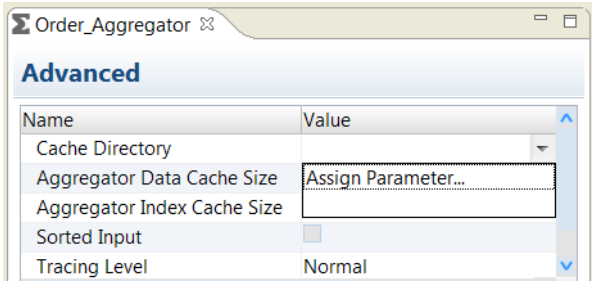
After you bind a mapping parameter to the transformation parameter, you can createa parameter set or a parameter file to override the mapping parameter value at run time. Run the mapping from the Developer tool or the command line. Specify the parameter set or parameter file to use for the mapping run.

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.

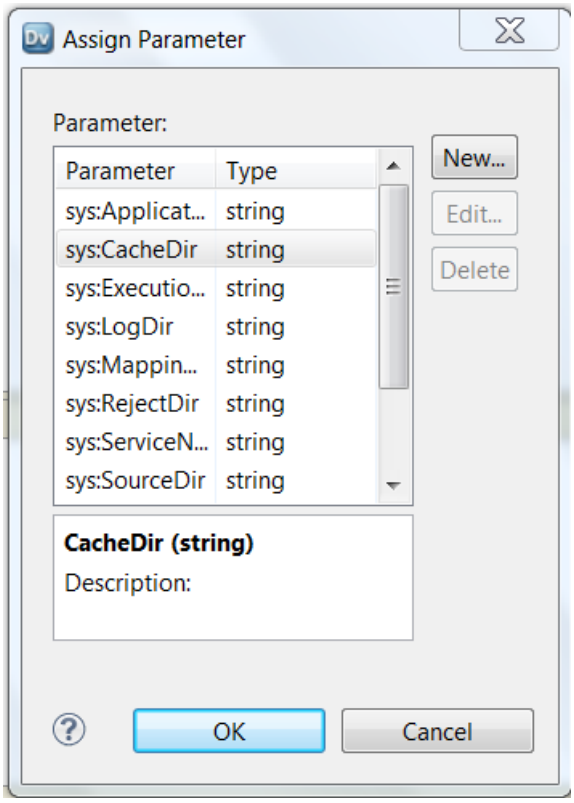
1. Navigate to the field or property that you want to update.
2. Click the selection arrow in the **Value** column.

   If you can parameterize the property, the **Assign Parameter** option appears.
   The following image shows the **Assign Parameter** option for the Cache Directory:



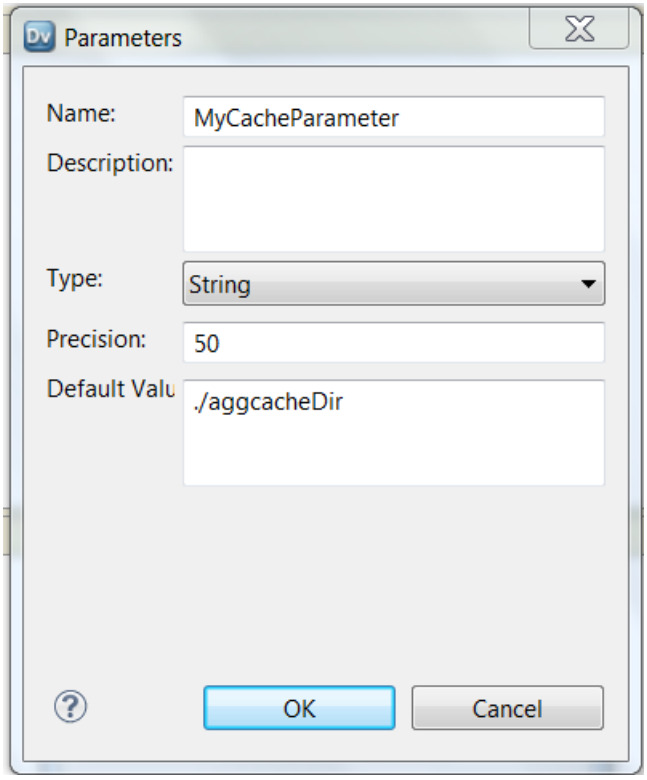
3. Click **Assign Parameter**.

   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:



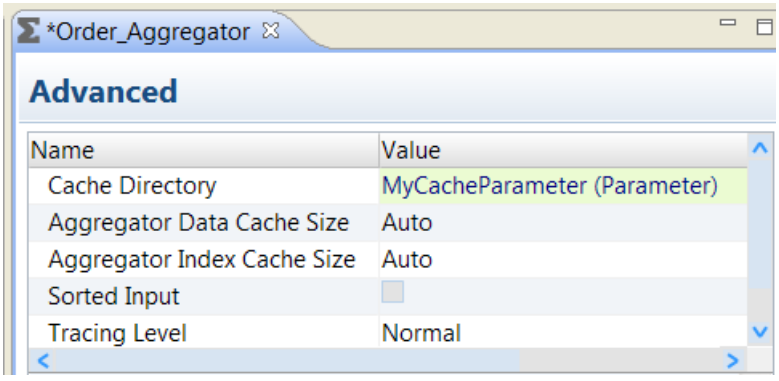
4. To create a parameter, click **New**.
5. Enter the parameter name, the type, the precision, and the default value.

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



6. Click **OK**.

   The parameter name appears in the transformation property.
   The following image shows MyCacheParameter in the Aggregator transformation cache directory:



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.

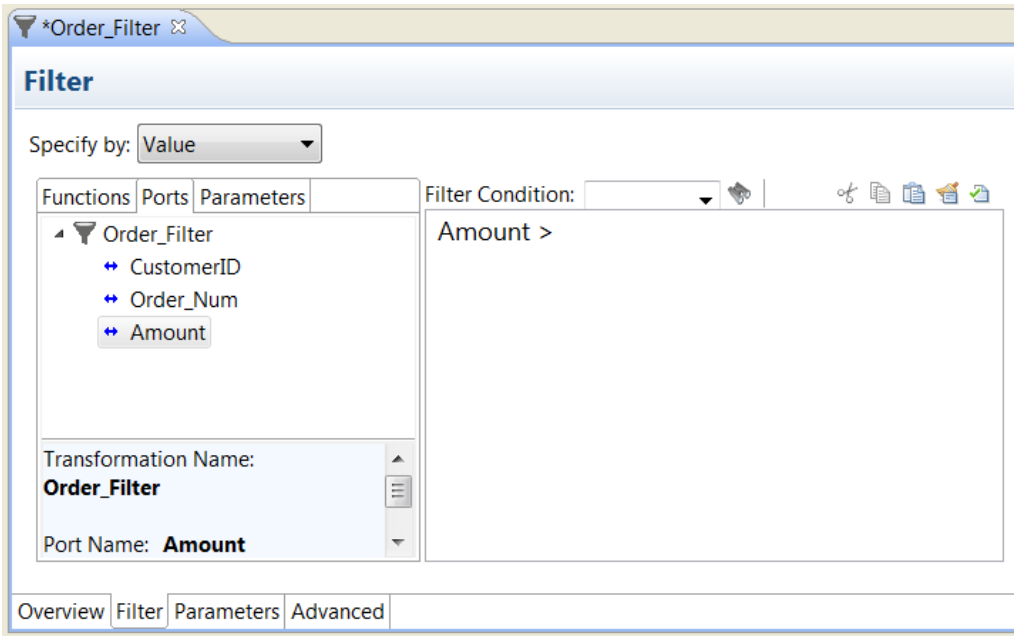
1. In the Filter transformation, click the **Filter** tab.

   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:



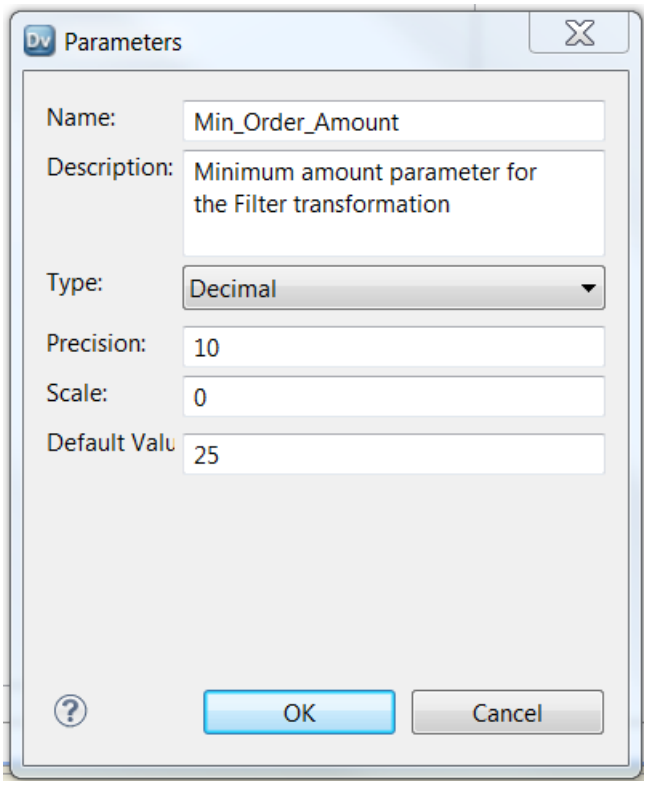
5. Click the **Parameters** tab in the Expression Editor.

   The Expression Editor lists the system parameters and the user-defined parameters.
6. Click **Manage Parameters** to add a parameter.

   The **Parameters** dialog box appears.
7. Click **New**.

   A dialog box appears with default parameter values.
8. Enter the parameter name, parameter type, precision, and default value.

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

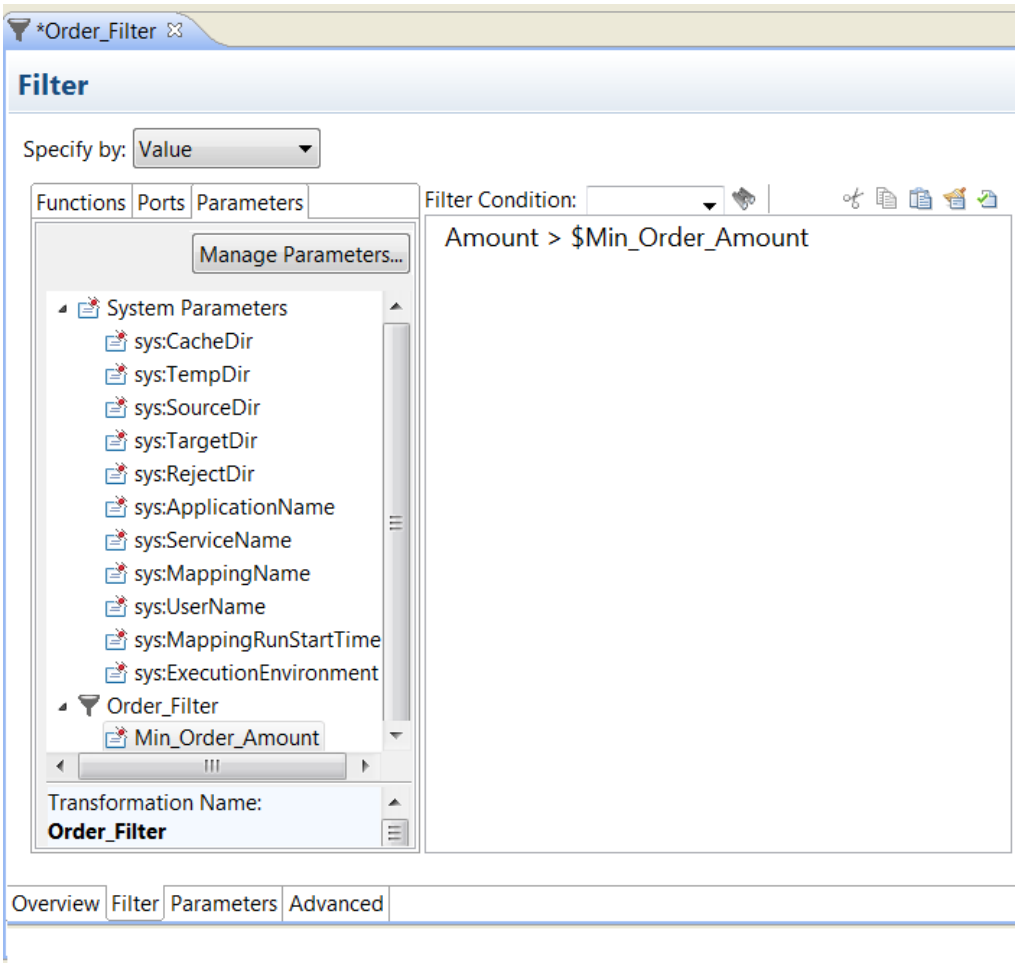


9. In the Expression Editor, click **OK**

   The parameter that you created appears in the parameter list.

10. 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

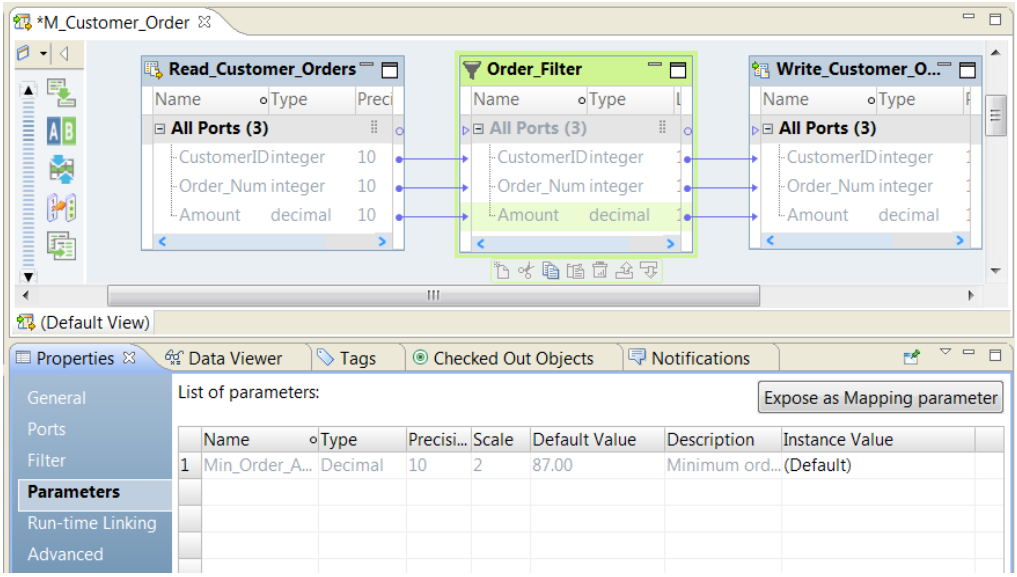
After you add a reusable transformation to a mapping, you can expose the transformation parameter as a mapping parameter. When you expose a reusable transformation parameter as a mapping parameter, you create a mapping parameter from the transformation parameter. The mapping parameter has the same name and type as the transformation parameter. The mapping parameter uses the instance value of the reusable transformation parameter.

You can click **Expose as Mapping Parameter** one time for a transformation parameter. If you click **Expose as Mapping Parameter** and the transformation parameter is already bound to a mapping parameter, the Developer tool does not change the mapping parameter. The Developer tool does not create another mapping parameter, and it does not update the mapping parameter default value. After you create a mapping

parameter, multiple objects might use the mapping parameter. If you need to change the mapping parameter default value, change the value in the mapping or change it at run time.

1. Open the mapping. Select the mapping in the Outline view.

   The **Parameters** tab appears in the **Properties** view.
   The following image shows the **Parameters** tab for a Filter transformation:



2. 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.

3. To update a mapping parameter, select the parameter from the **Outline** view.

   You can change the default mapping parameter value. You can also add mapping parameters on the mapping **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.

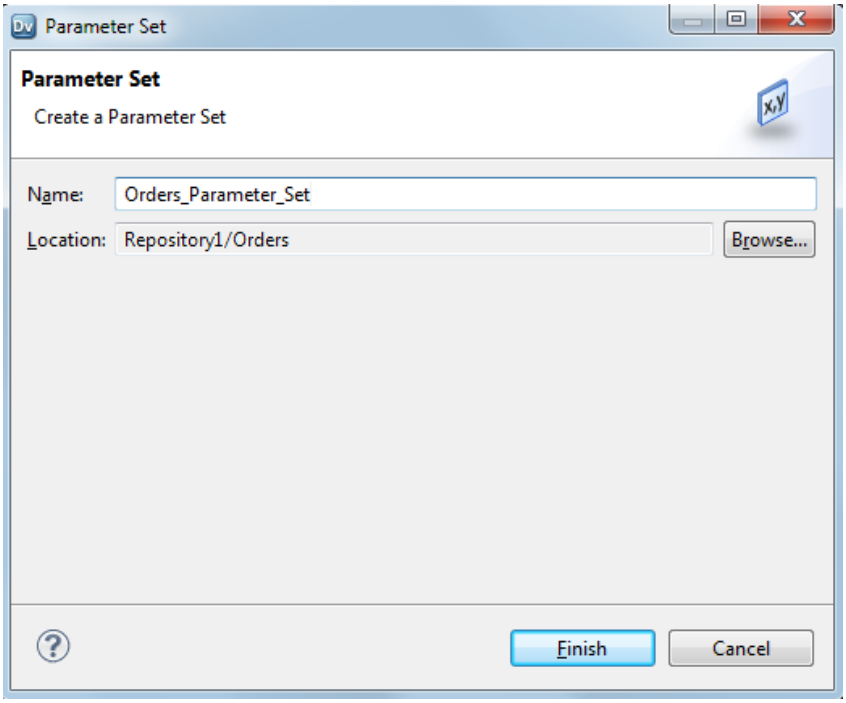
c. To use the transformation parameter default value, click **Use Default**.

Creating a Parameter Set

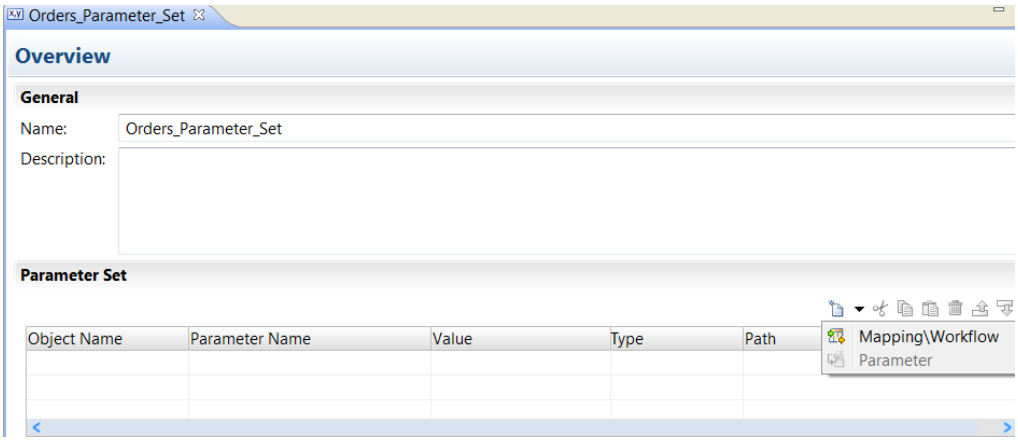
Create a parameter set that you can use to change the runtime context for mappings and workflows.

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**.

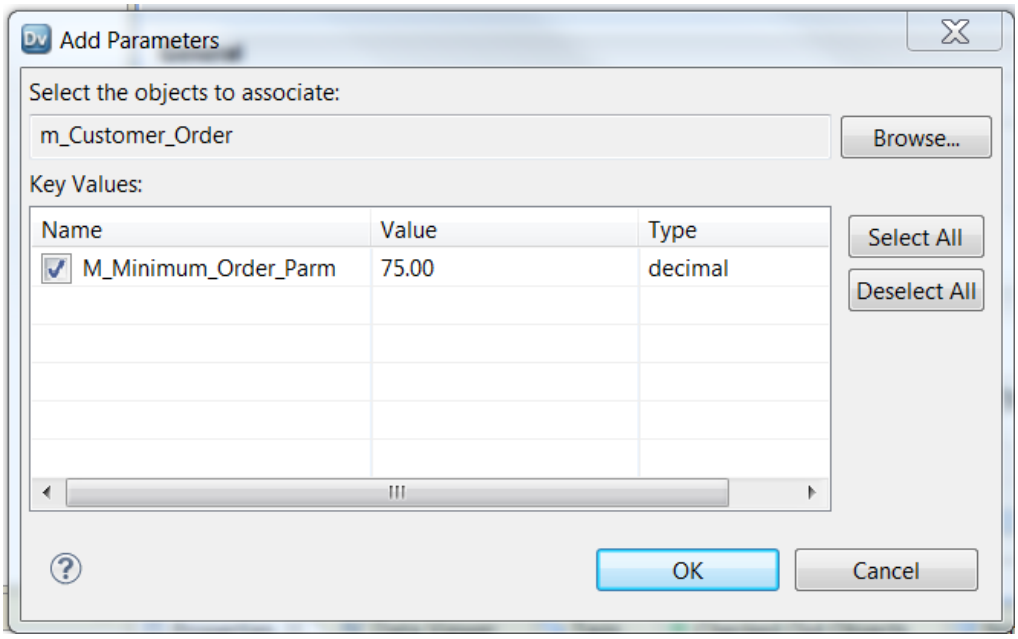


5. In the **Add Parameters** dialog box click **Browse** to find the mapping or workflow that contains the parameters you need to include in the set.

   A list of mappings and workflows appears.

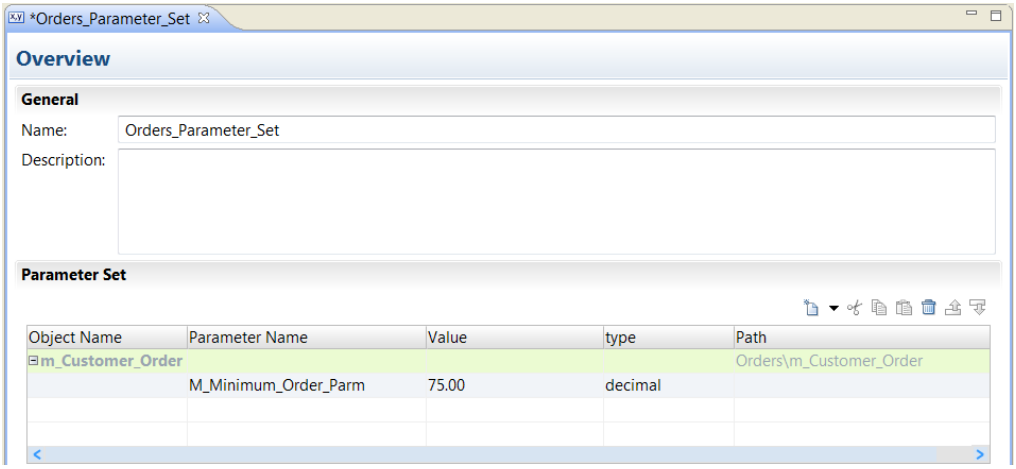
6. Select a mapping or a workflow and click **OK**.

   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.

How to Run a Mapping with Parameters

You can run a mapping with parameters from the Developer tool or the command line.

To run a mapping with parameters from the Developer tool, run the mapping using advanced options. In the advanced options, specify the parameters. To run the mapping from the command line, specify the mapping and the parameter set or parameter file.

How to Run a Mapping with Parameters from the Developer Tool

Before you run a mapping with parameters from the Developer tool, validate the parameters. To validate the parameters, resolve the mapping parameters and validate the mapping. Then run the mapping using advanced options. In the advanced options, specify whether you want to use the default parameter values, a parameter set, or a parameter file.

Resolving and Validating Parameters

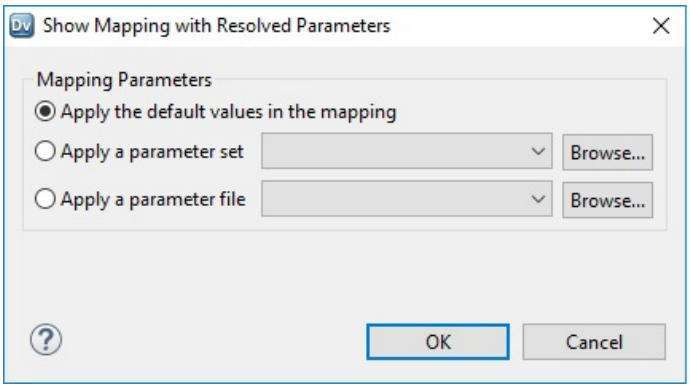
You can resolve and validate mapping parameters to ensure that the Data Integration Service can read and process the parameters at run time. The mapping must be valid before you resolve the parameters. When you

resolve the mapping parameters, the Developer tool generates an instance of the mapping that shows how the Data Integration Service resolves the parameters at run time.

If you run a mapping multiple times using different parameter sets or parameter files, validate the mapping with resolved parameters each time to make sure that the parameter sets or parameter files are compatible with the mapping.

1. In the Developer tool, right-click a mapping in the editor or the Object Explorer view. Select **Show Mapping with Resolved Parameters**.

   The **Show Mapping with Resolved Parameters** dialog box appears.



2. Select one of the following mapping parameters options:
   - Apply the default values in the mapping. The Data Integration Service applies the default parameter values configured in the mapping.
   - Apply a parameter set. The Data Integration Service applies the parameter values configured in the parameter set.
   - Apply a parameter file. The Data Integration Service applies the parameter values configured in the parameter file.
3. Click **OK**.

   The Developer tool generates a run-time instance of the mapping that shows the resolved parameters. The run-time instance of the mapping appears in a new tab.
4. Right-click the run-time instance of the mapping and select **Validate**.

   If errors appear in the **Validation Log** view, fix the errors and validate the mapping again.
5. To run the mapping with the selected parameters, right-click the run-time instance of the mapping and click **Run**.

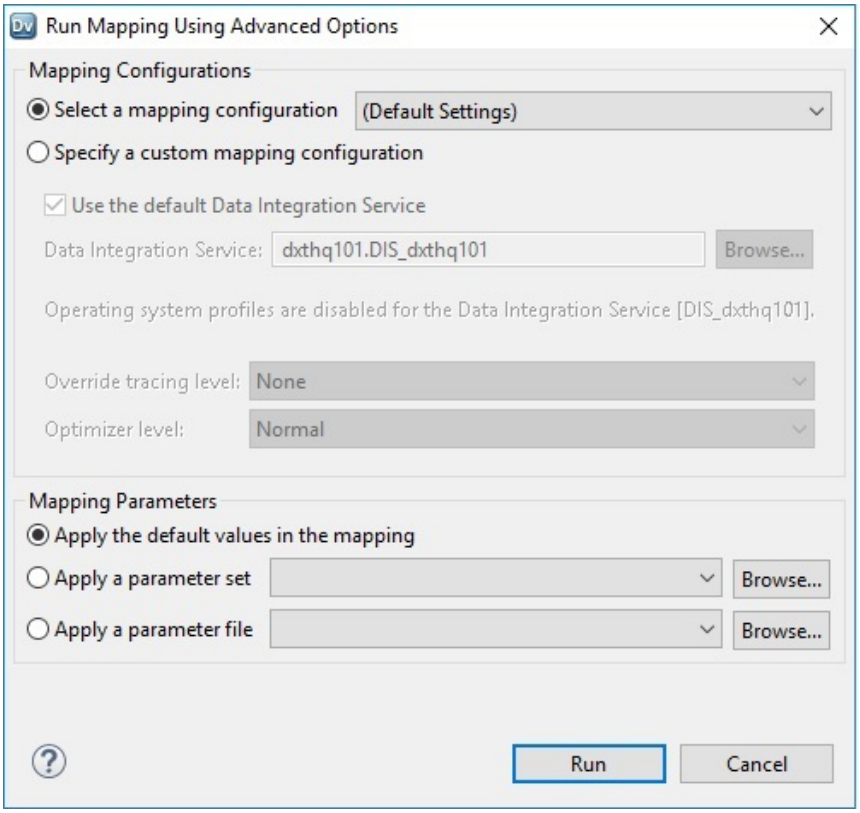
Running a Mapping with Parameters

To run a mapping with parameters from the Developer tool, run the mapping using advanced options. In the Developer tool, you can run a mapping using the default parameter values in the mapping, a parameter set, or a parameter file.

If you run a mapping instance where the parameters are resolved, you cannot specify different parameters to run the mapping. The mapping runs using the parameters that are resolved in the mapping.

1. In the Developer tool, right-click a mapping in the editor or the Object Explorer view. Select **Run Mapping Using Advanced Options**.

   The **Run Mapping Using Advanced Options** dialog box appears.



2. Select one of the following mapping configuration options:
   - Select a mapping configuration. Choose a reusable mapping configuration to run the mapping.
   - Specify a custom mapping configuration. Select the Data Integration Service, operating system profile, override tracing level, and optimizer level.
3. Select one of the following mapping parameters options:
   - Apply the default values in the mapping. The Data Integration Service applies the default parameter values configured in the mapping.
   - Apply a parameter set. The Data Integration Service applies the parameter values configured in the parameter set.
   - Apply a parameter file. The Data Integration Service applies the parameter values configured in the parameter file.

How to Run a Mapping with Parameters from the Command Line

To run a mapping with parameters from the command line, you must deploy the mapping as an application. After the application is deployed, run the mapping. Specify the mapping and the parameter set or parameter file.

Running a Mapping with a Parameter Set

To run a mapping with a parameter set from the command line, you must deploy the mapping as an application and specify the parameter set in the application. Run the deployed mapping and specify the parameter set.

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 the parameter set that you want to use.

After the application is deployed, use the infamcd `addParameterSetEntries` command to add parameter set entries. Use the infacmd `updateParameterSetEntries` command to update parameter set entries.

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

Running a Mapping with a Parameter File

To run a mapping with a parameter file from the command line, you must deploy the mapping as an application. Run the mapping and specify the parameter file. Use the infacmd ms RunMapping command. The -pf argument specifies the parameter file name.

For example, the following command runs the mapping MyMapping using the parameter file "MyParamFile.xml":

```
infacmd ms RunMapping -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a
MyApplication -m MyMapping -pf 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.

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

CHAPTER 4

Where to Assign Parameters

This chapter includes the following topics:

- Where to Assign Parameters Overview, 82
- Parameters for Compression Formats, 86
- Parameters in Custom Queries for Hive Sources, 87
- Parameters in Custom Queries for Relational Sources, 88
- Parameters in Expressions, 89
- Expression Parameters, 91
- Expression Parameter Examples, 92
- Parameters for Fields and Property Values, 95
- Parameters for Relational Table Resources, 96
- Parameters in SQL Statements, 96
- Parameters for Port Lists, 99

Where to Assign Parameters Overview

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 assign a parameter to a property, the option to assign a parameter appears when you configure the property value.

For example, you can create a Read transformation in a mapping. The Read transformation is a non-reusable transformation that you create based on a physical data object. You can assign parameters to the transformation properties, or you can assign parameters to the physical data object.

You cannot nest user-defined parameters within a parameterized source. If the source data object is parameterized, you cannot expose a user-defined parameter as a mapping parameter to override the parameter values at run time. The mapping uses the default value instead.

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

Object	Field
All transformations	Link resolution order
Association transformation	Cache file directory Cache file size
Address Validator transformation	Casing style Default country Geocode data type Global maximum field length Line separator Maximum result count Optimization level Standardize invalid addresses
Aggregator transformation	Cache directory Expression elements Port expression Group By
Bad Record Exception transformation	Lower Threshold Upper Threshold
Case Converter transformation	Reference table.
Consolidation transformation	Cache file directory Cache file size
Customized data object	Connection Data object Owner SQL Query elements Table name
Customized data object read operation	Custom query Filter condition Join condition PreSQL PostSQL
Customized data object write operation	PreSQL PostSQL Update override
Decision transformation	Decision script.
Duplicate Record Exception transformation	Cache file directory Lower Threshold Upper Threshold

Object	Field
Expression transformation	Expression elements Port expression Port selector Sort key list. Windowing only.
Filter transformation	Filter condition elements Filter condition. Full expression.
Flat file data object	Compression codec Compression format Control file directory Control file name Connection name Default scale Flat file delimiter Merge file directory Source file directory Source file name Output file name Output file directory Reject file directory Target directory
Joiner transformation	Cache directory Join condition elements Port selector
Key Generator transformation	Cache file directory Cache file size
Labeler transformation	Reference table
Lookup transformation	Custom query. Relational only.
Lookup transformation, excluding the physical data objects for the lookup source	Data object. Nonreusable transformation. Dynamic port rules. Nonreusable transformation. Lookup condition. Full expression, nonreusable transformation. Port selector. Nonreusable transformation.
Mapping	Hive version Run-time environment Maximum parallelism
Match transformation	Cache directory on the Match Output tab Cache directory on the Match Type tab Index directory on the Match Type tab Persistence method Threshold

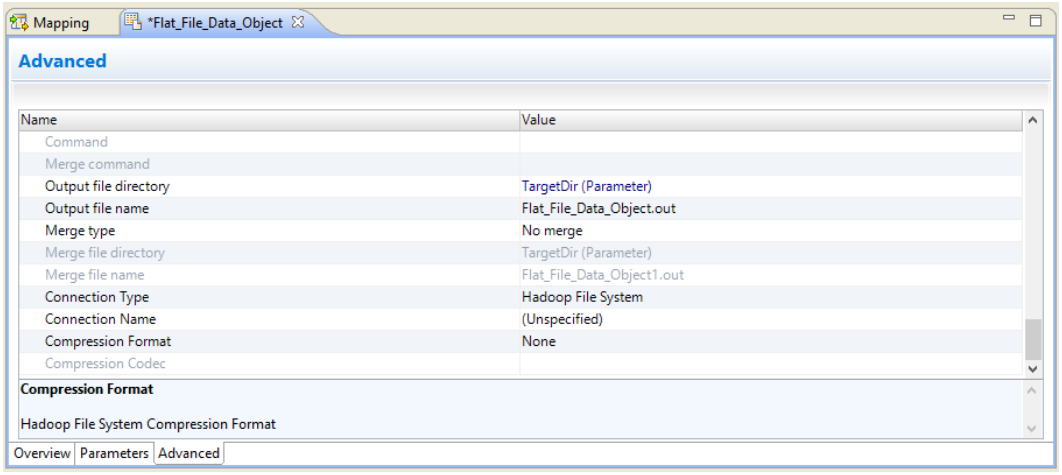
Object	Field
Nonrelational data object	Connection
Rank transformation	Cache directory Expression elements Port expression Group by ports Rank port
Read transformation	Connection Custom query. Relational only. Data object Filter condition. Relational only. Join condition. Relational only. Owner name. Relational only. PreSQL. Relational only. PostSQL. Relational only. Resource/table name. Relational only.
Relational Data Object	Filter condition elements Join condition elements PreSQL query elements PostSQL query elements SQL override elements
Router transformation	Group filter condition elements. Group filter condition. Full expression.
Sorter transformation	Sort key Group by Work directory
SQL transformation	Connection
Standardizer transformation	Reference table
Token Parser transformation	Reference table
Update Strategy transformation	Update strategy expression elements. Update strategy expression. Full expression.
Write transformation	Data object Link resolution order PreSQL. Relational only. PostSQL. Relational only. Reject directory Reject file name Update override. Relational only.

Parameters for Compression Formats

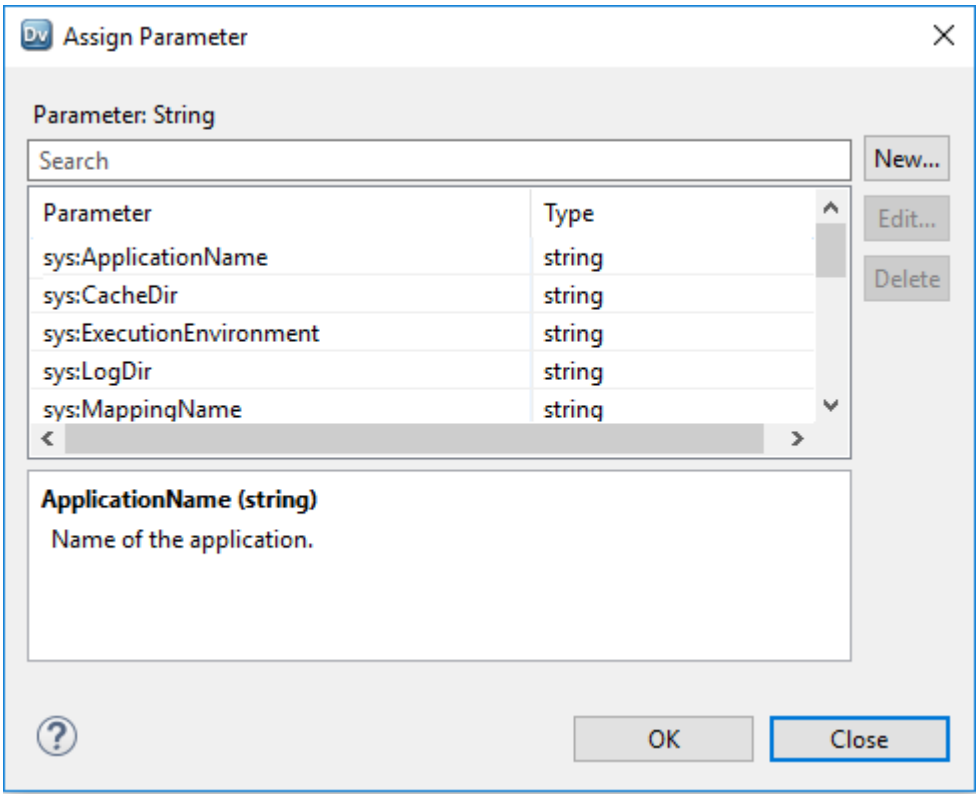
You can assign a parameter to the compression format and the compression codec for an HDFS flat file target.

You must assign a parameter to the compression format before you can assign a parameter to the compression codec. If you assign a parameter to the compression format, you must assign a parameter to the compression codec.

The following image shows the advanced properties for an HDFS flat file where you can configure the compression format and the compression codec:



The following image shows the dialog box that you can use to assign a parameter to the compression format and the compression codec:

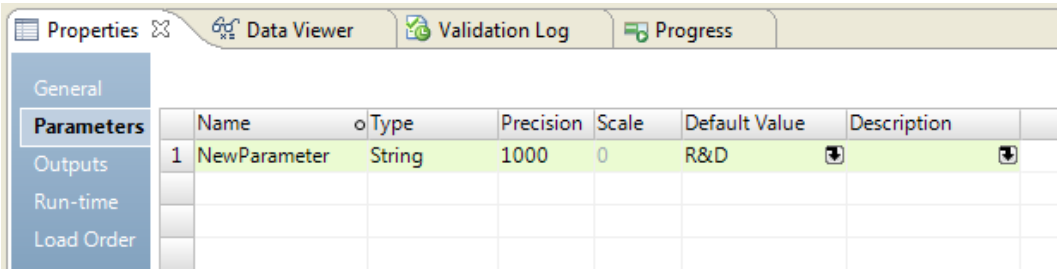


Parameters in Custom Queries for Hive Sources

When you use a string parameter in a SQL override, a join expression, or a filter query for a Hive source, you need to add quotes around the parameter reference if the parameter represents a literal value. You can use single or double quotes. This requirement is for Hive sources in mappings that run in the native execution environment or in the Hadoop execution environment.

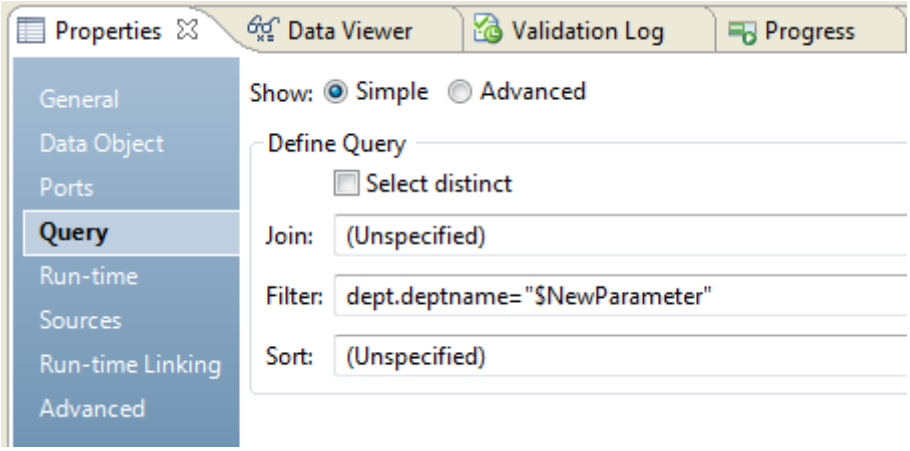
For example, you need to create a filter that selects Hive source rows with a specific department name. You create a string parameter that represents the department name. You assign a default value of R&D for the department name parameter.

The following image shows the string parameter:



When you use the parameter in a filter query for a Hive source, you must include quotes around the parameter name. Otherwise the mapping fails at run time with a SQL parser error.

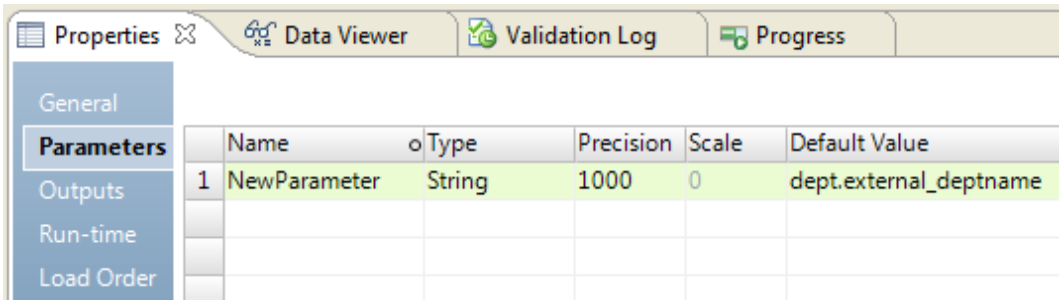
The following image shows the filter query for the Hive source on the **Query** view of the **Properties** tab:



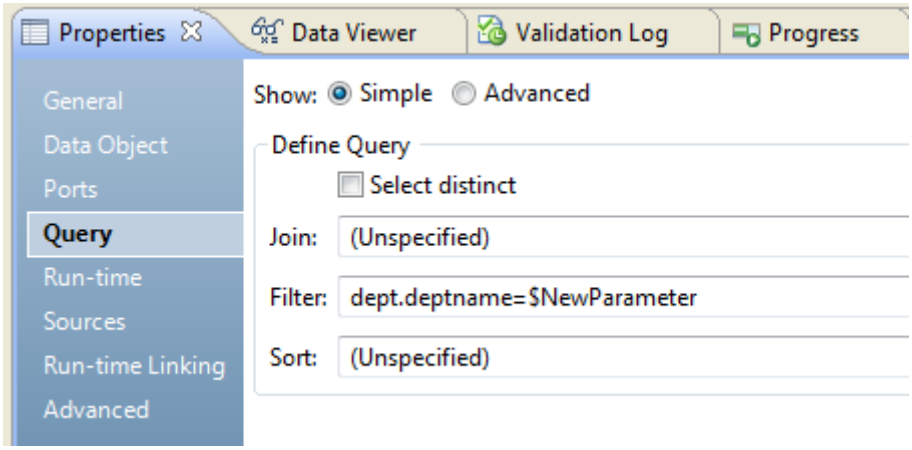
Note: By default, the Expression editor does not add the quotes around the parameter. You must manually add them.

You do not need to add single or double quotes around the parameter name if the parameter contains a column name or a sub query name.

The following image show a string parameter with a default value that is a column name:



The following image shows a filter query that uses the parameter:

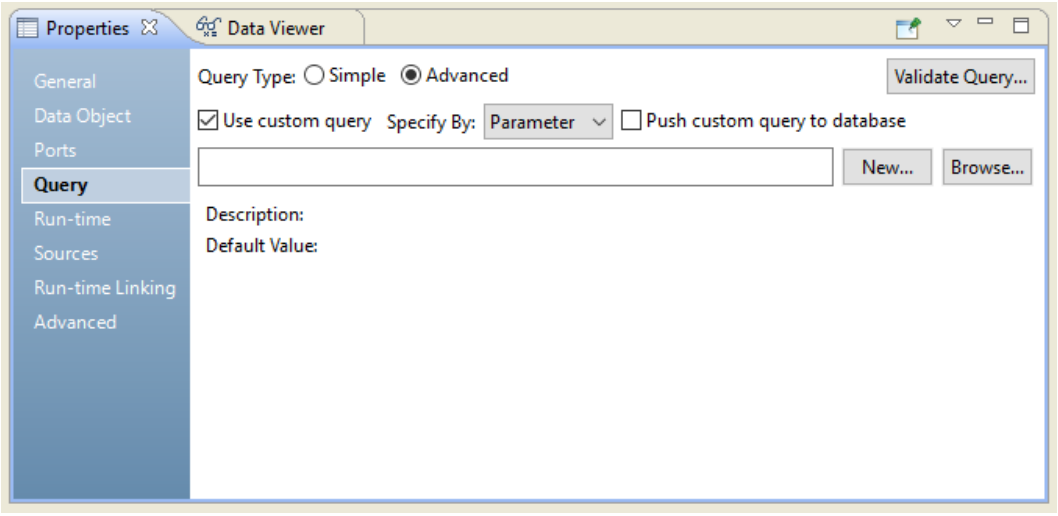


Parameters in Custom Queries for Relational Sources

You can assign a parameter to the custom query for a customized data object read operation, or a relational data object in a Read transformation.

When you configure the custom query, you must create an advanced query and specify the query using a parameter.

The following image shows where you can parameterize the custom query:

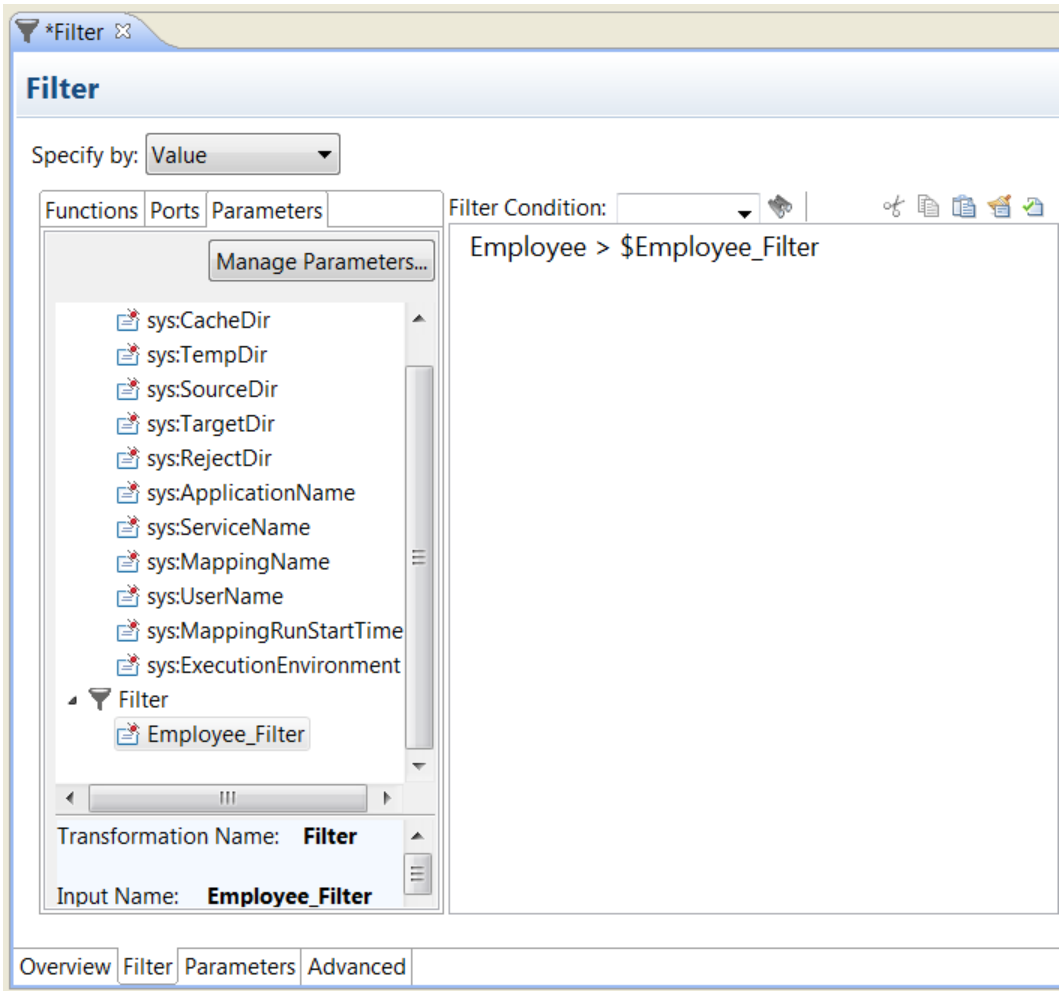


Parameters in Expressions

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:



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( Input_Port,$Scale_Param )
```

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 the following transformation fields:

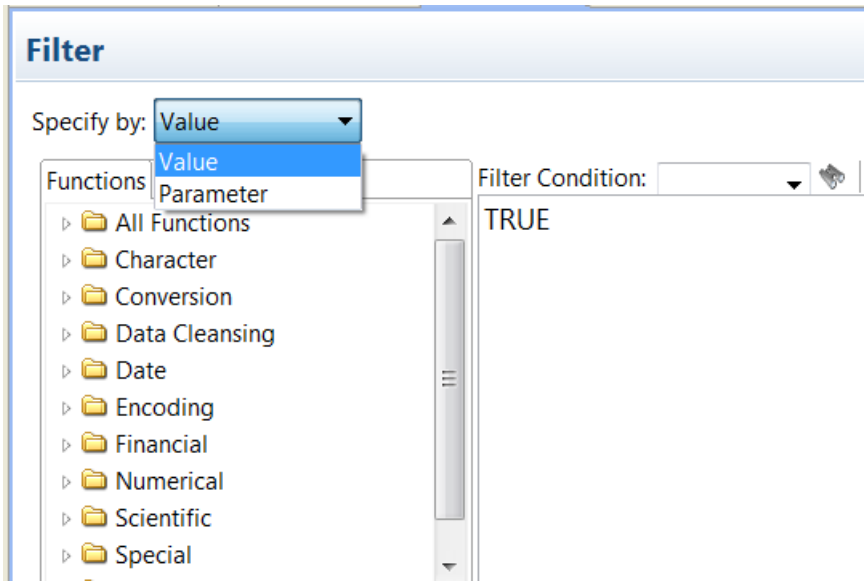
- Aggregator. Port expression.
- Expression. Port expression.
- Filter. Filter condition.
- Join. Join condition.
- Lookup. Lookup condition.
- Rank. Port expression.

If you configure an expression parameter in a filter, join, or lookup condition in a Filter, Join, or Lookup transformation, the expression parameter must evaluate to TRUE or FALSE.

If you configure an expression parameter in the port expression for an Aggregator, Expression, or Rank transformation, the port must contain only the expression parameter. The result value of the parameter must match the port data type. For example, if the port data type is a string, the expression must evaluate to a string.

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

For example, 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. For example, you might create expression parameters with different port names and operators in a dynamic mapping. The expression parameter cannot contain other parameters.

If the expression parameter is a transformation parameter, you can bind the transformation parameter to a mapping parameter. You can override the mapping parameter at run time.

Restrictions for Port Expressions

If you use expression parameters for port expressions, consider the following restrictions:

- If you create the expression parameter as a mapping parameter, you cannot validate the default value of the parameter unless the parameter is used as a port expression in a transformation.
- The mapping fails if the default value of the expression parameter is not valid.
- The mapping fails if the return value of the expression parameter does not match the data type of the port.

Expression Parameter Examples

This section contains examples that demonstrate how you can use expression parameters in transformations.

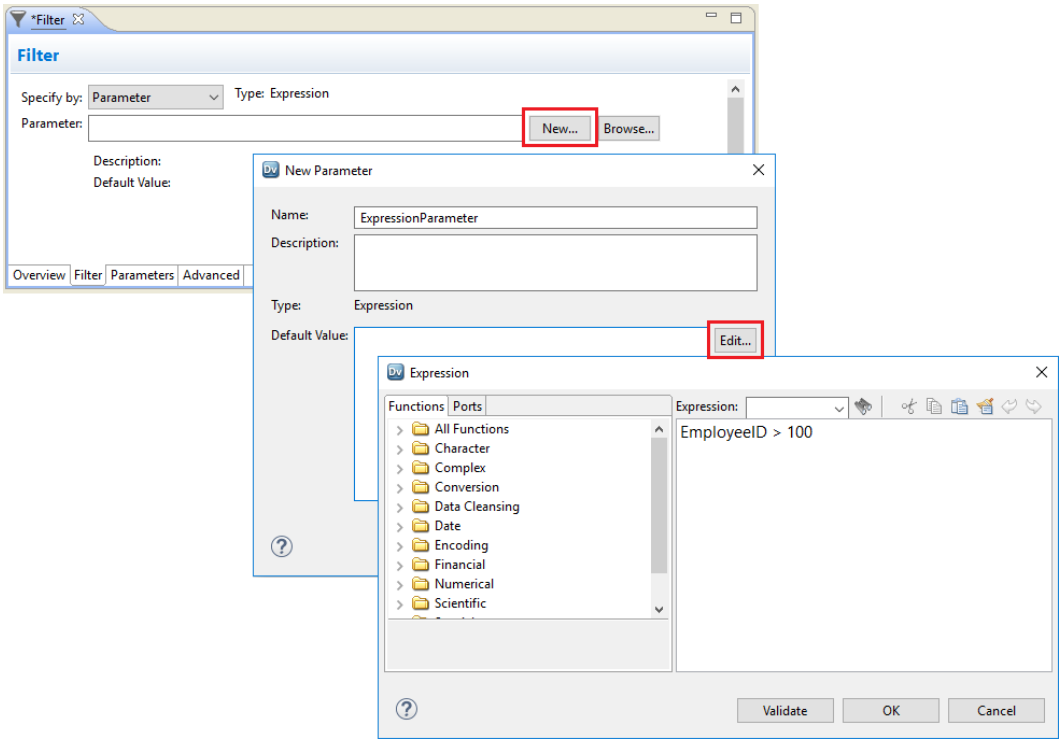
Example. Expression Parameters in a Filter Condition

Use expression parameters to parameterize the filter condition in a Filter transformation.

For example, you might use an expression parameter for a filter condition with the following default value:

```
EmployeeID > 100
```

The following image shows where you can configure the expression parameter:

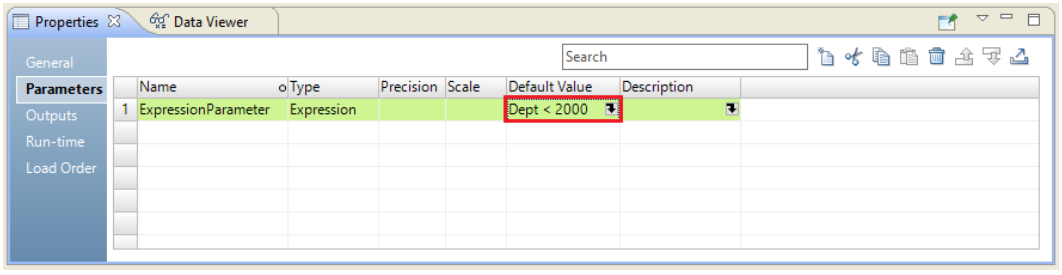


If the Filter transformation is reusable, you can bind the transformation parameter to a mapping parameter. You can assign a different default value to the mapping parameter and use the mapping parameter in non-reusable transformations in the mapping.

For example, you can edit the mapping parameter and configure the following default value:

```
Dept < 2000
```

The following image shows the default value of the mapping parameter:



Example. Expression Parameters in an Expression Transformation

Use an expression parameter to parameterize the port expression in an Expression transformation.

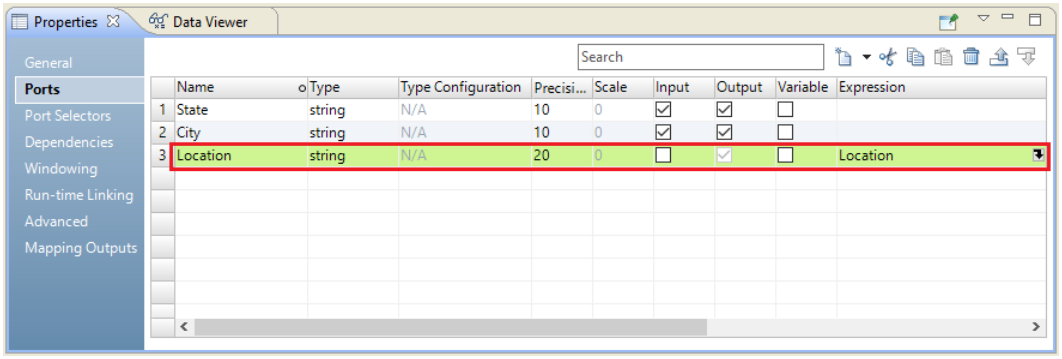
For example, you might want to use an Expression transformation to concatenate data on states and cities.

The following table lists the data that you might transform:

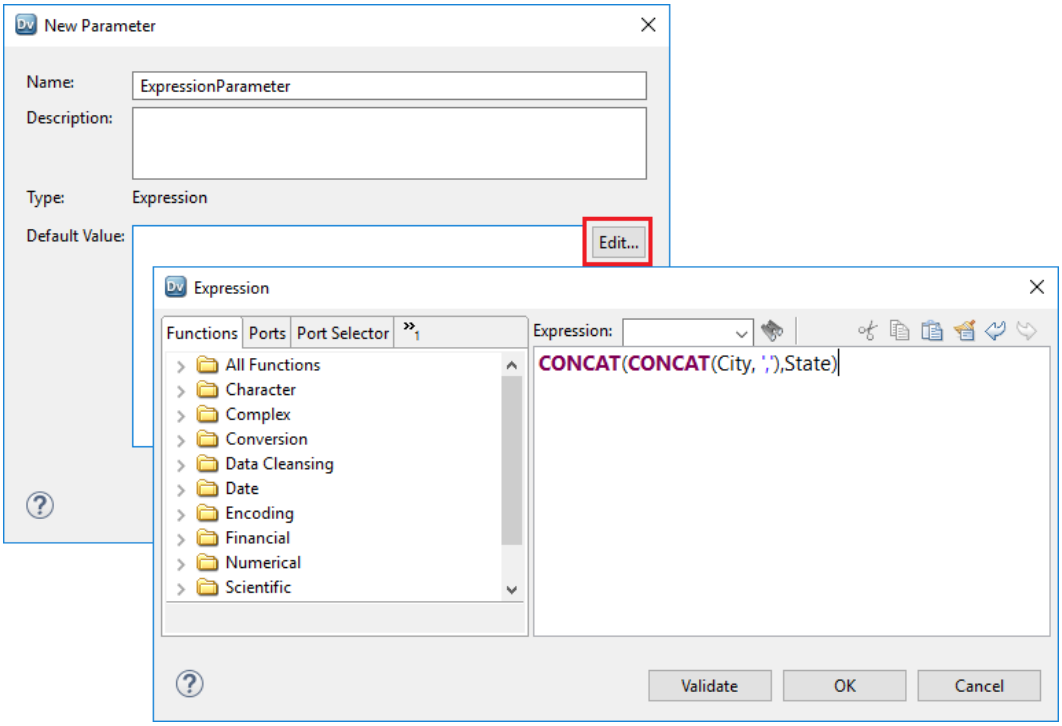
State	City
CA	Sacramento
TX	Austin
NY	Albany

You want to concatenate the data using the expression `CONCAT(CONCAT(City, ','),State)`. In the Expression transformation, configure an output port `Location` and assign an expression parameter to the port expression.

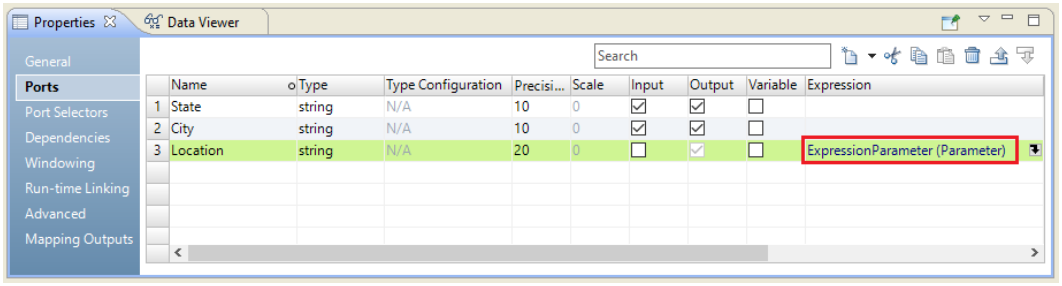
The following image shows the output port `Location` in the Expression transformation:



The following image shows where you can configure the new expression parameter:



The following image shows where the expression parameter appears:



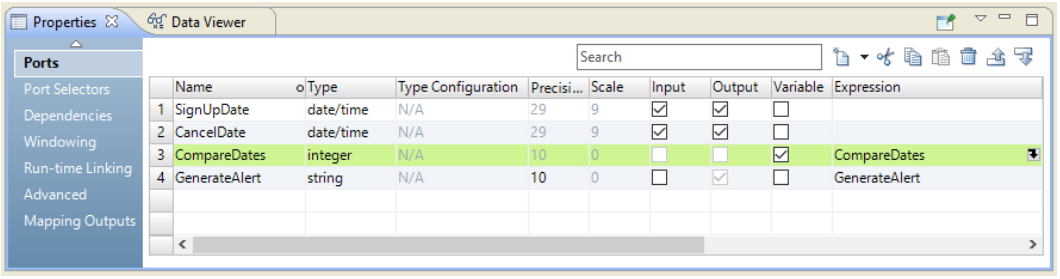
When you run the mapping, the output port uses the expression parameter to write data using the format `City,State`, such as `Sacramento,CA`.

Example. Nesting Parameters in Port Expressions

To nest expression parameters, you can assign expression parameters to different ports and use the ports in other expression parameters.

For example, you might have data on dates that customers opened and closed accounts with your company. You want to inspect the data to ensure that the dates in your system are valid. If the dates are not valid, you want to issue a warning.

To inspect the data, you can create an Expression transformation and configure the following ports:



The input port `SignUpDate` is the date that a customer opened an account. The input port `CancelDate` is the date that a customer closed an account.

The variable port `CompareDates` returns -1, 0, or 1 depending on whether the value in the port `CancelDate` is earlier or later than the value in the port `SignUpDate`. The port `GenerateAlert` returns 'Warning' if the value in the port `CompareDates` is 1.

For the variable port `CompareDates`, you can configure an expression parameter with the following default value:

```
DATE_COMPARE(SignUpDate,CancelDate)
```

To nest the expression parameter that you use for the variable port `CompareDates` in the expression parameter for the output port `GenerateAlert`, you can configure an expression parameter with the following default value:

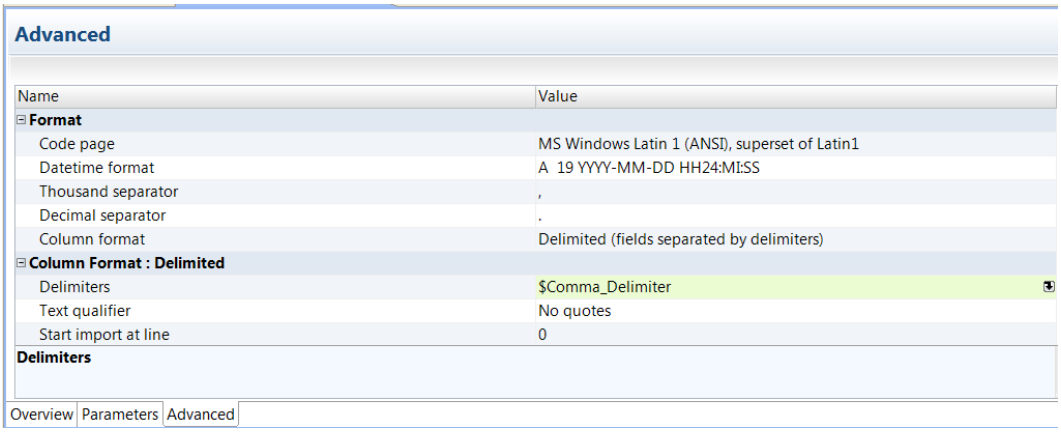
```
IIF(CompareDates=1,'Warning')
```

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:



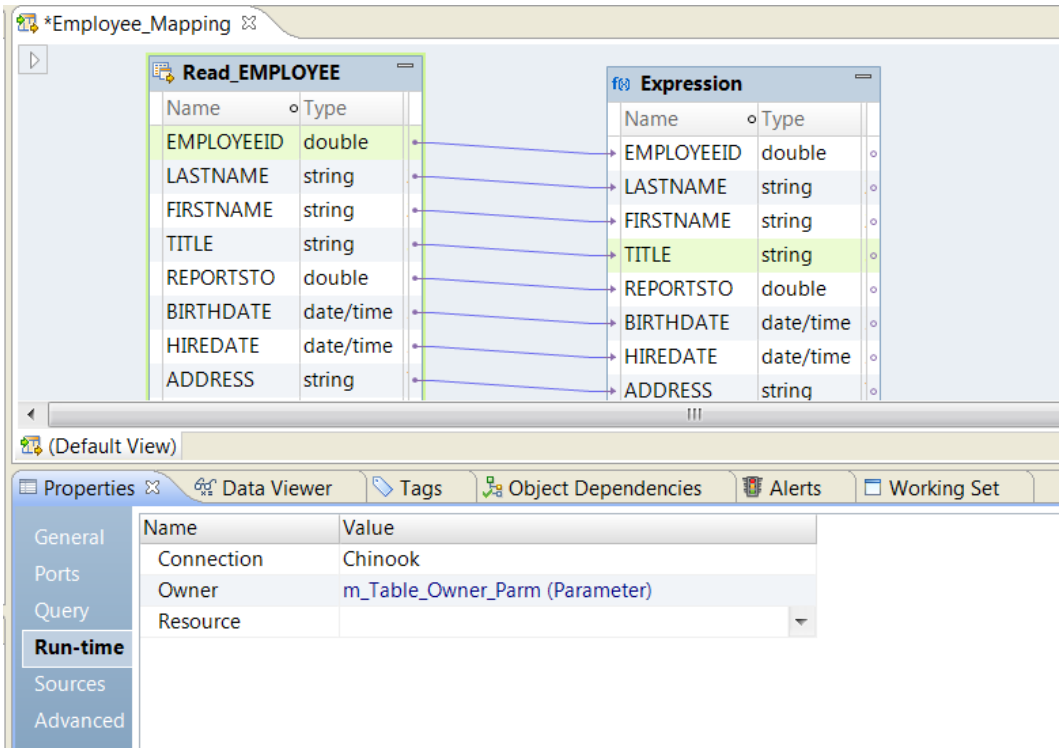
Parameters for Relational Table Resources

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.

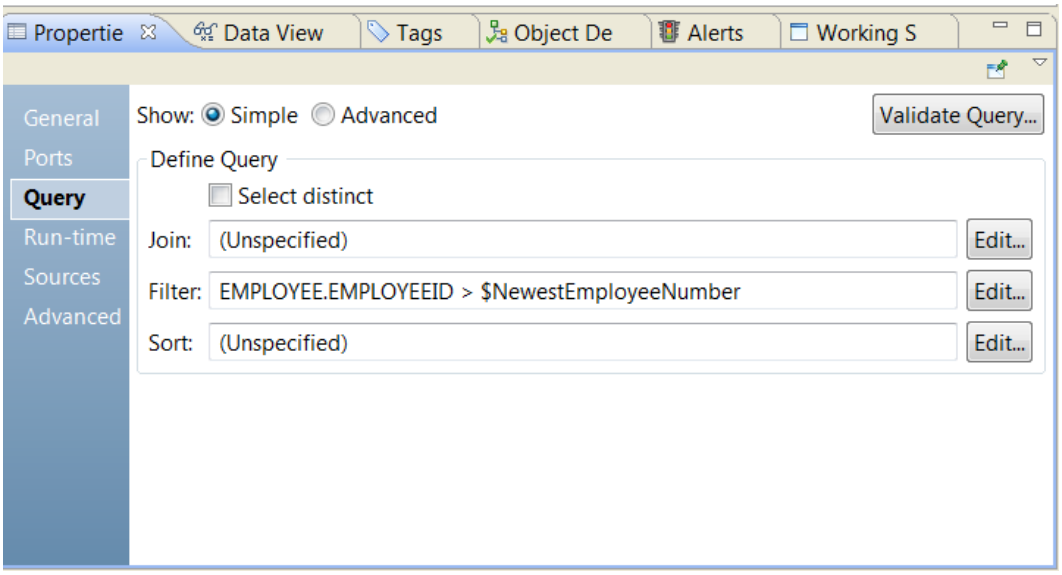
The following image shows where to assign the parameters for the connection, the resource name, and the table owner for the Read transformation:



Parameters in SQL Statements

You can include parameters in SQL statements that you add to relational data objects, customized data objects, or to Read and Lookup transformations that use relational or customized data objects.

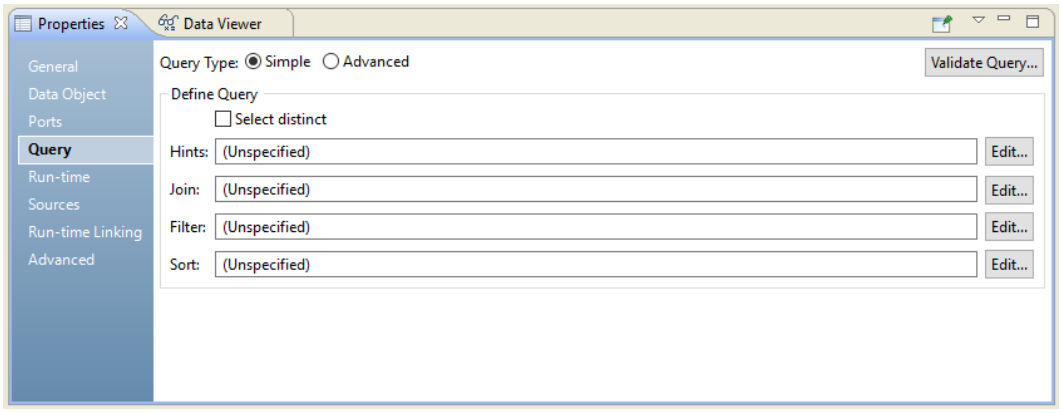
The following image shows how you can parameterize an SQL query that reads from a relational source:



Parameters in Filter and Join Conditions

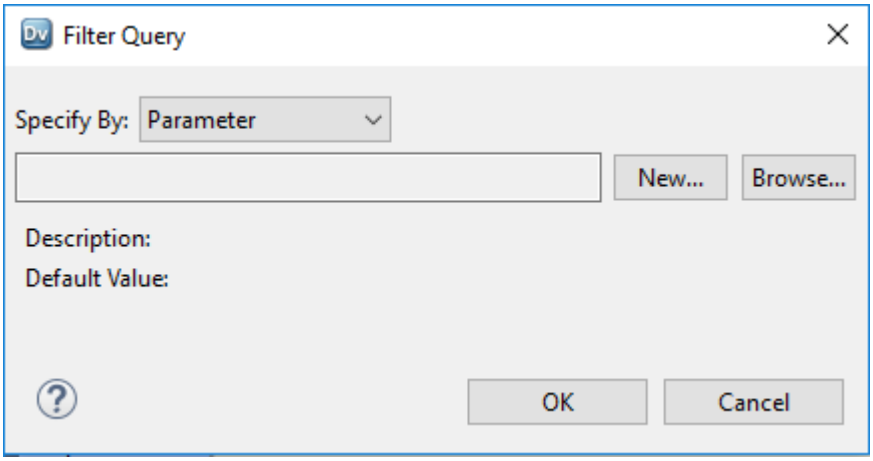
You can assign a parameter to the SQL statement that you add to the filter or join condition. You can assign the parameter in a customized data object read operation; or in a relational data object in a Read transformation.

The following image shows where you can assign a SQL parameter to a filter or join condition:



When you edit the join or the filter condition, you can use a parameter to specify the condition.

For example, the following image shows the dialog box that you can use to assign a parameter to the SQL statement when you edit the filter condition:



String Parameters in SQL Statements

When you define a string parameter in an SQL statement, you must add single quotes (' ') to the parameter in the query. If single quotes are part of the default value instead of the query, the Data Integration Service escapes the data in each parameter with single quotes and adds additional single quotes for every single quote that appears in the default value.

Example - Single Quotes in the Query

For example, you have an SQL statement with a date parameter $date_parm. The SQL statement appears as the following expression:

```
select * from <table_name> where <date_port> > $date_parm
```

Since the parameter $date_parm is a string, add single quotes to the parameter in the query. The following expression shows the query where single quotes are added to the parameter:

```
select * from <table_name> where <date_port> > '$date_parm'
```

If the default value of the parameter $date_parm is `01/31/2000 00:00:00`, the following expression shows how the Data Integration Service expands the query:

```
select * from <table_name> where <date_port> > '01/31/2000 00:00:00'
```

Example - Single Quotes in the Default Value

You use the same SQL statement with a date parameter $date_parm. The SQL statement appears as the following expression:

```
select * from <table_name> where <date_port> > $date_parm
```

You add single quotes to the default value `'01/31/2000 00:00:00'`. The following expression shows the expanded query:

```
select * from <table_name> where <date_port> > ''01/31/2000 00:00:00''
```

Since single quotes are part of the default value, the Data Integration Service escapes the data with additional single quotes. Because the string parameter contains double quotes in the expanded query, the query might fail or produce no result.

Tips for Using Parameters in SQL Statements

You can use tips to use parameters in SQL statements more effectively.

- For a string parameter, use single quotes when you define the parameter in the query.
- Do not use single quotes for a parameter if the parameter is not a string. You might get unexpected results.
- For a string parameter, if the SQL query and the default value of the parameter do not have single quotes, you can add an upstream Filter transformation in the mapping. In the Filter transformation, edit the filter condition to include single quotes around the parameter.
- 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:

  ```
  SELECT $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 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`

CHAPTER 5

Mapping Outputs

This chapter includes the following topics:

- Mapping Outputs Overview, 100
- User-Defined Mapping Outputs, 101
- System-Defined Mapping Outputs, 103
- Persist and Bind Mapping Outputs in Deployed Mappings, 104
- Mapping Outputs In Mapplets, 109
- Mapping Outputs in Logical Data Objects, 113
- How to Bind Mapplet Outputs to Mapping Outputs, 113

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.

The mapping returns each mapping value when the mapping completes. You can define multiple mapping outputs in the same mapping.

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

You can bind mapping outputs to mapping parameters. The mapping uses the persisted output from a previous run as the value for a parameter the next time the mapping runs. You cannot directly use the mapping output in a transformation, but you can use the parameter that is bound to the mapping output. If there is no persisted output value, the parameter uses a default value.

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 do not have to define a system-defined mapping output.

If the mapping runs as a Mapping task in a workflow, you can persist the mapping output in the repository or bind the mapping output to a workflow variable.

For information on mapping outputs in Mapping tasks, see the "Mapping Task" chapter in the *Informatica Developer Workflow Guide*.

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.

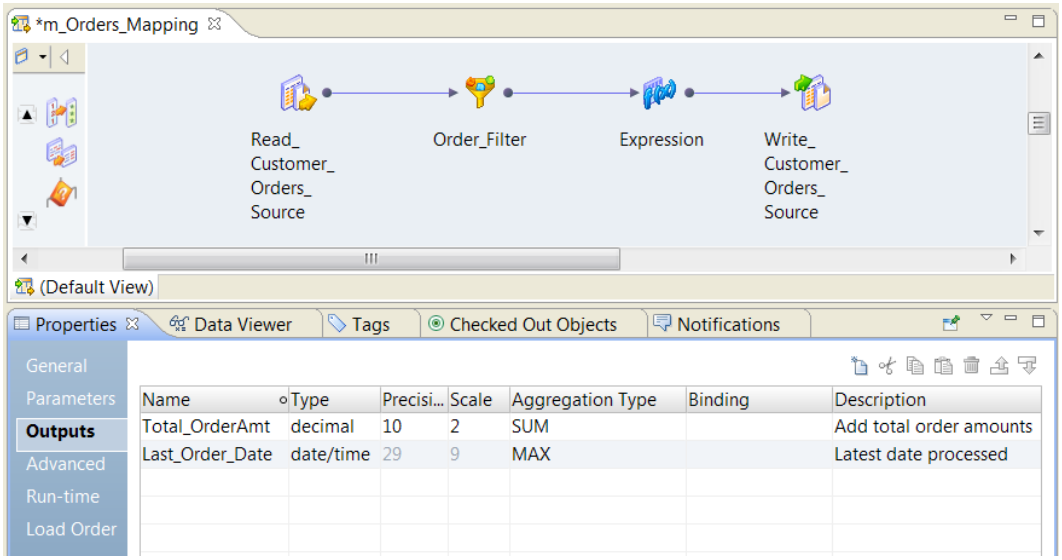
Define the mapping output in the **Outputs** tab of the Properties view. Configure a name for the mapping output, a data type for the result, and indicate what type of aggregation to perform to return a result.

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

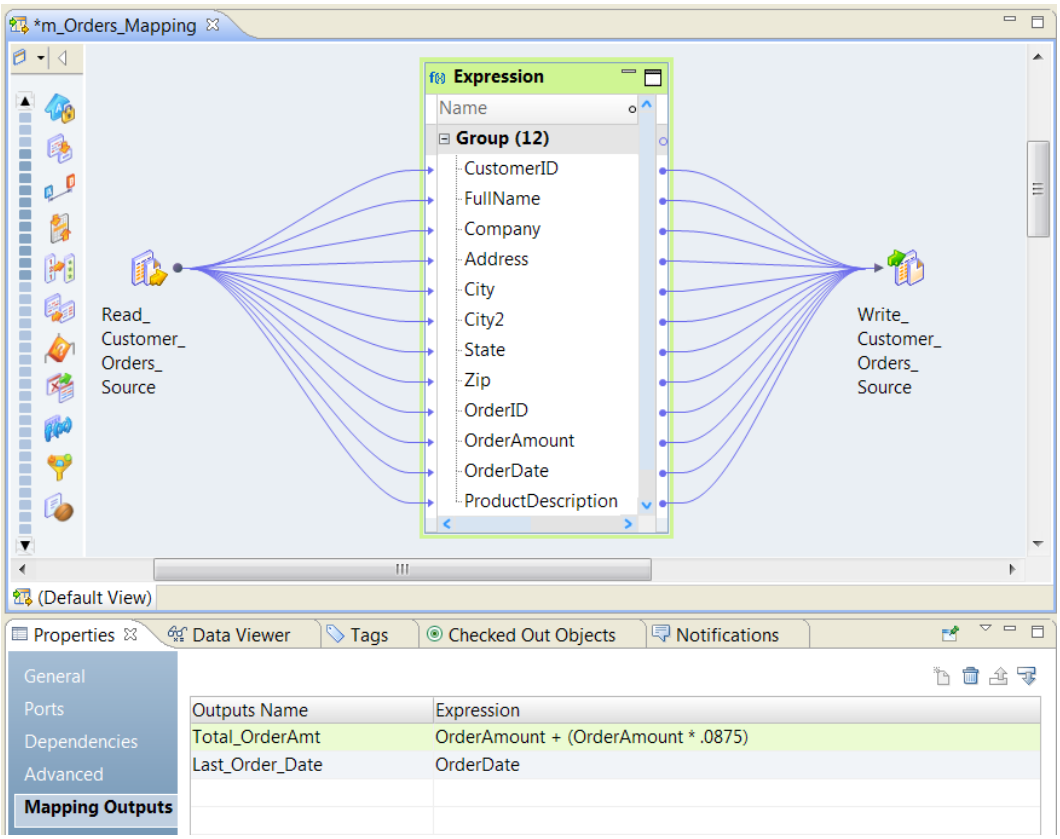
The description of the mapping output.

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.

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.

You can bind system-defined mapping outputs to mapping parameters and workflow variables.

For information on mapping outputs in workflows, see the "Mapping Task" chapter in the *Informatica Developer Workflow Guide*.

Persist and Bind Mapping Outputs in Deployed Mappings

Persist mapping outputs to save output values in the repository. For mappings deployed to run in the native environment or on the Spark engine, you can bind the output to a mapping parameter to use the value in subsequent runs of the same mapping.

When you persist and bind mapping outputs, define a filter query for the Read transformation and a mapping output expression in an Expression transformation. The filter query checks new data against the persisted output value. The mapping output expression aggregates an output value for the mapping.

When you define the mapping parameter, assign a default value for the parameter. When you bind a mapping output to a mapping parameter, the persisted output overrides the default value unless you configure a value for the same parameter in a parameter set or parameter file. If you define a parameter value in a parameter set or parameter file, that value will be used for the parameter.

You must set a run-time instance name to persist mapping output values. To set a run-time instance name, use the -RuntimeInstanceName option for the infacmd ms runMapping command. For more information, see the *Informatica Command Reference*.

A use case for binding is to persist the latest ID number that the mapping processes. The next time that the mapping runs, the mapping parameter uses the value of the last ID processed. The mapping can filter the parameter source rows to include only the rows with an order date greater than the last order date processed.

How to Configure and Bind Mapping Outputs

When you configure mapping outputs, define the mapping outputs at the mapping level and configure the expressions to aggregate at the transformation level. Bind the persisted output to a mapping parameter.

To configure and bind mapping outputs in deployed mappings, perform the following steps:

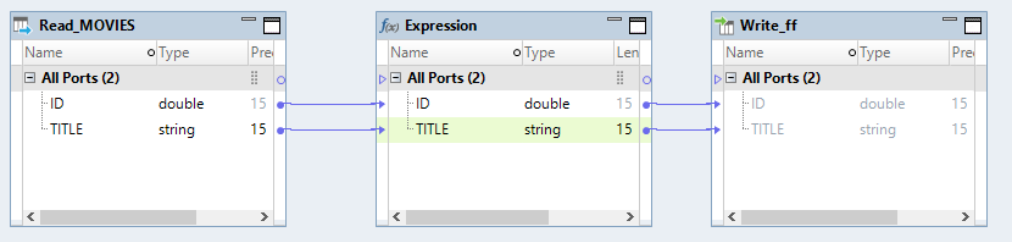
1. Create a mapping with an Expression transformation and configure mapping parameters.
2. Define a mapping output to persist in the repository and set the aggregation type used to calculate the output.
3. Configure a mapping output expression to aggregate an output for each row that the mapping processes.
4. Define a filter query in the Read transformation to read data incrementally from the data source based on a parameterized field and its value.
5. Bind the mapping output to a parameter to use the persisted value in subsequent mapping runs.

Step 1. Create the Mapping and Configure Mapping Parameters

Create a mapping with an Expression transformation that writes to a local flat file target. Define input parameters for the mapping.

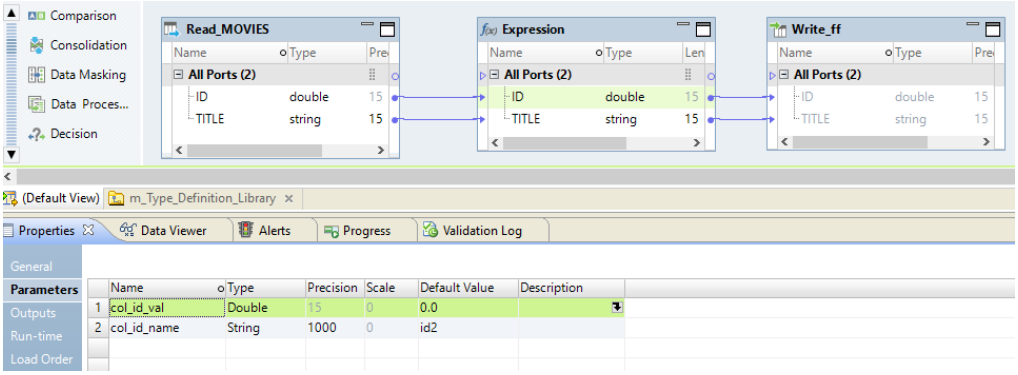
1. Create a mapping to process data from a MOVIES file. Use a Read transformation to read from the data source.
2. Include an Expression transformation immediately after the source. The Expression transformation includes an expression to aggregate the output value.
3. Use a Write transformation to write the result to a local flat file target.

   The following image shows the mapping:



4. Click the editor to access the mapping properties.
5. Open the **Parameters** view.
6. Create a parameter that will accept the persisted mapping output value called col_id_val. Set a default value for the parameter.
7. Create a parameter for the name of the ID column in the source table called col_id_name.

   The following image shows the mapping parameters:



8. Create a parameter set that includes the col_id_name parameter. Set the value for the parameter to *MOVIES.ID*.

Step 2. Define the Mapping Output and Aggregation Type

Define mapping outputs in the Outputs tab of the mapping Properties. Each mapping output definition describes what type of aggregation to perform and the data type of the results. The Output column includes both user-defined and system-defined mapping outputs.

1. After you create a mapping, click the editor to access the mapping properties.
2. Open the **Outputs** view.
3. Create a new mapping output.

   The Developer tool creates a mapping output with default field values.

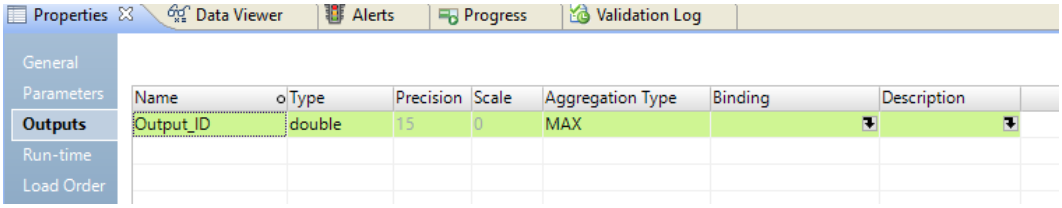
4. Change the name that identifies the mapping output to Output_ID.
5. Select a numeric mapping output type. Enter the precision and scale.
6. Choose the aggregation type for the mapping output as MAX to store the largest order ID processed.

   When you select the aggregation type, you can choose to calculate the sum of the values that the mapping processes or find the minimum or maximum value. Default is SUM.

   Note: When you persist mapping outputs, the aggregation type is also used to compare the previously persisted value to the newly calculated value.
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 the mapping output Output_ID:



Step 3. Configure the Expression Transformation

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

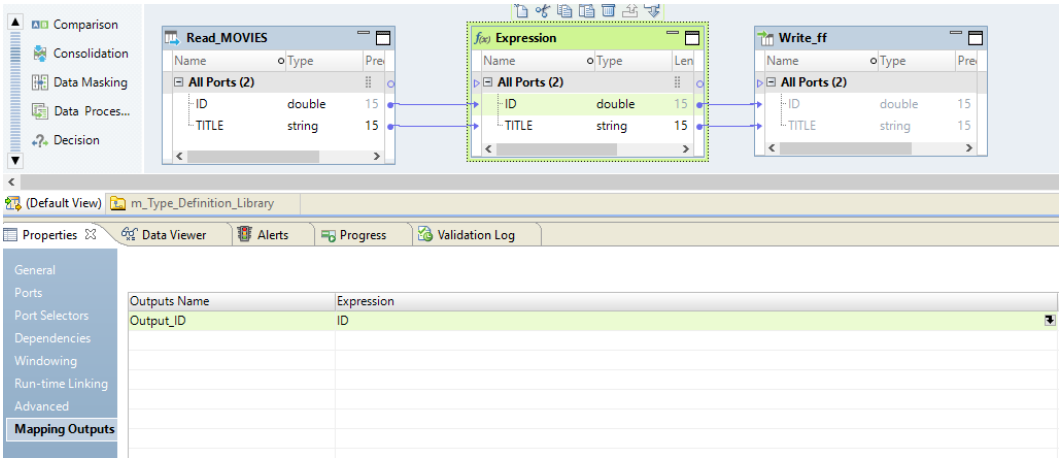
1. Select the Expression transformation in the mapping editor to view the transformation properties.
2. Open the **Mapping Outputs** view.
3. Create a mapping outputs expression.

   The Developer tool creates an output expression with an 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.
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. If the expression contains parameters, the parameter values can be set in a parameter file or parameter set.
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 expression.
7. Click **File** > **Save** to save the Expression transformation.

The following image shows the saved mapping output expression. The expression uses only the port 'ID' to calculate the mapping output.



Step 4. Define the Filter Query

In the Read transformation, configure a filter query to read data incrementally based on a parameterized field and its value.

1. Select the Read transformation in the mapping editor to view the transformation properties.
2. Open the **Query** view.
3. Select the **Edit** button for the **Filter** property.
4. Configure a filter to find values that are greater than the persisted value using the following form: $<input port parameter> **>** $<persisted value parameter>

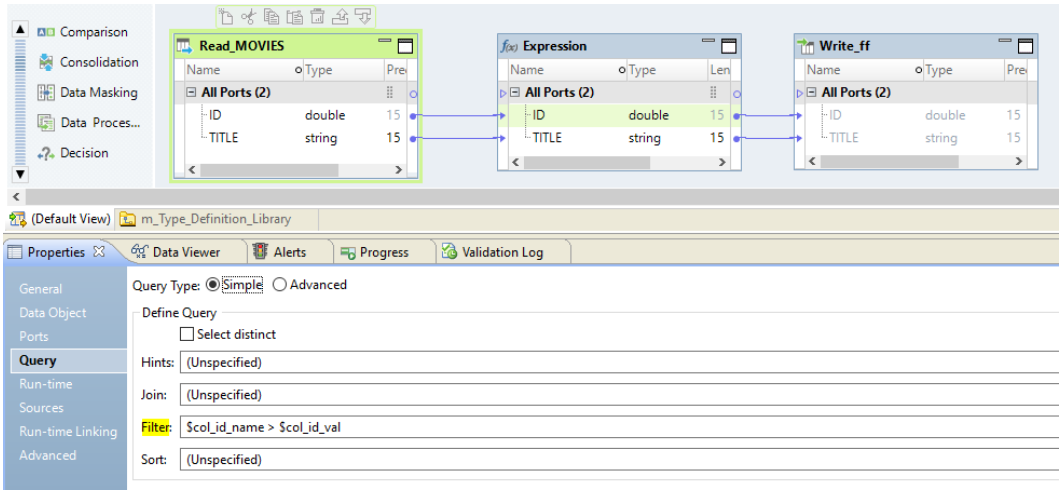
   Input port parameter is the mapping parameter col_id_name for the ID column from the source relational table.

   Persisted value parameter is the mapping parameter col_id_val that you bind with the persisted mapping output value.

   If you assign a value to the col_id_val parameter in a parameter set or parameter file, the query uses that assigned value for the parameter. Otherwise the query uses the persisted value, if one exists, or the default value.
5. Click **OK** to save the filter query.

   The filter query appears in the **Filter** property.
6. Click **File** > **Save** to save the Read transformation.

The following image shows the configured filter query:

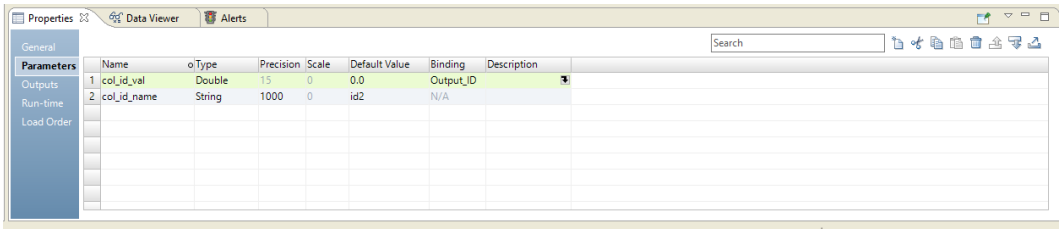


Step 5. Bind Mapping Outputs to Mapping Parameters

Bind the mapping output to the configured mapping parameter.

1. Click the editor to access the mapping properties.
2. Open the **Parameters** view.
3. For the parameter col_id_val that you defined to accept the persisted mapping output, use the drop down menu in the **Binding** column to select the mapping output Output_ID.

On subsequent runs of the mapping, the persisted value for Output_ID will be passed to the mapping parameter col_id_val. The following image shows the mapping parameters:



Persisted Outputs Maintenance

You can list and delete the persisted mapping outputs in the repository.

You can run the following infacmd ms commands for persisted mapping output values:

listMappingPersistedOutputs

Lists the persisted mapping outputs for a deployed mapping. The outputs are listed based on the name of the application and the run-time instance name of the mapping.

deleteMappingPersistedOutputs

Deletes all persisted mapping outputs for a deployed mapping. Specify the outputs to delete using the name of the application and the run-time instance name of the mapping. To delete specific outputs, use the option -OutputNamesToDelete.

For more information about infacmd, see the "infacmd ms Command Reference" chapter in the *Informatica® Command Reference*.

Rules and Guidelines for Persisted Mapping Outputs and Deployment

When you redeploy a mapping 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:

- The first time you deploy a mapping, you do not have to perform any additional tasks if a mapping has persisted mapping outputs.
- When you redeploy a mapping, the persisted mapping outputs are retained in the repository. You can run the infacmd ms deleteMappingPersistedOutputs to delete persisted outputs from the repository.
- The mapping outputs state is not backed up when you back up a mapping.
- If you rename a mapping output, you cannot use the persisted mapping output value from a previous deployed mapping run. If you recreate a mapping output with the same name, you can use the persisted output value.

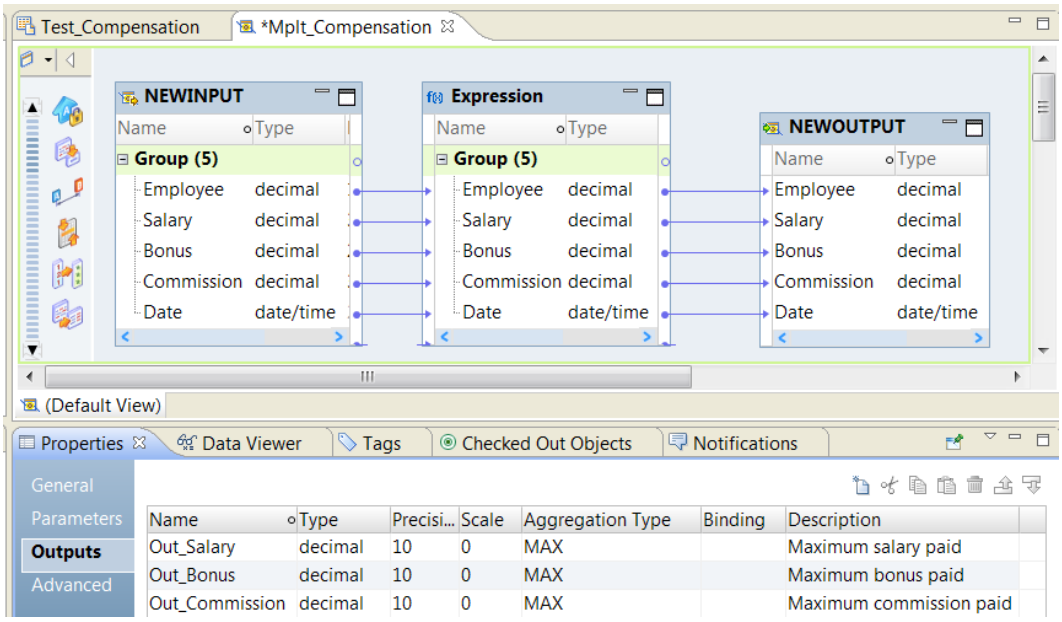
Mapping Outputs In Mapplets

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

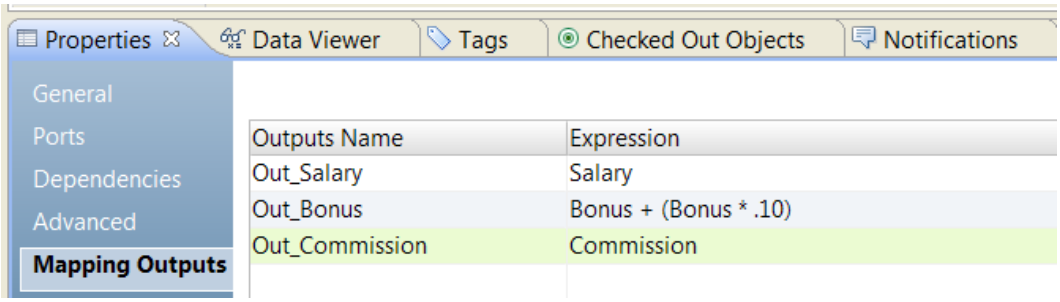
The name of an output from another mapplet to bind to this mapping output. The **Binding** field is blank unless the mapplet contains another mapplet that is returning the mapping output.

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:



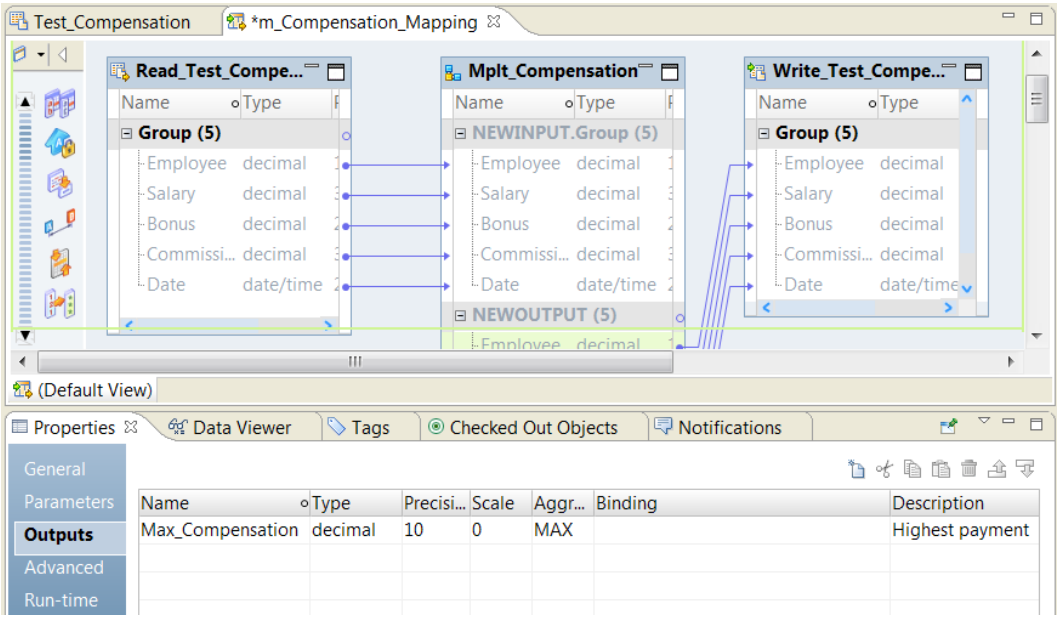
For this example, the Expression transformation aggregates the Salary and Commission port values. The Out_Bonus mapping output is an expression that includes the Bonus port value plus 10% of the Bonus.

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.

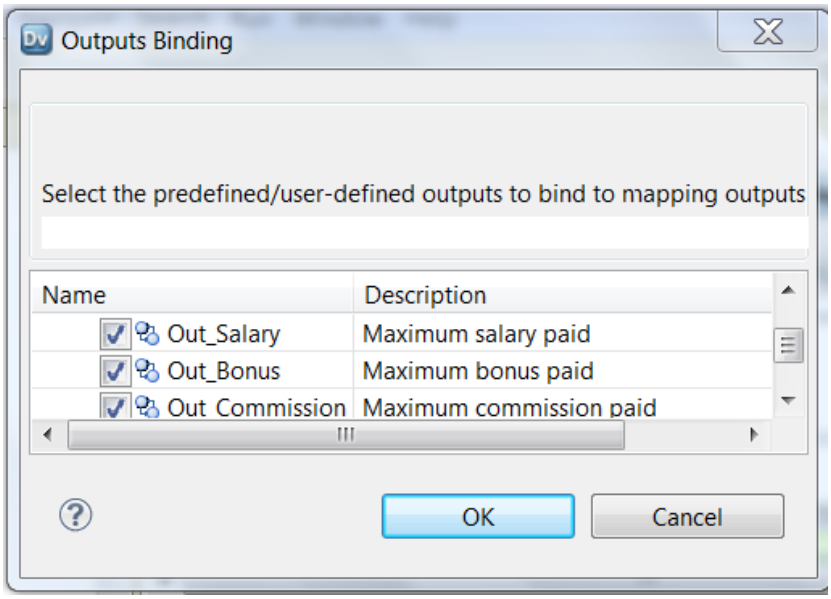
The following image shows the Max_Compensation mapping output at the mapping level:



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.

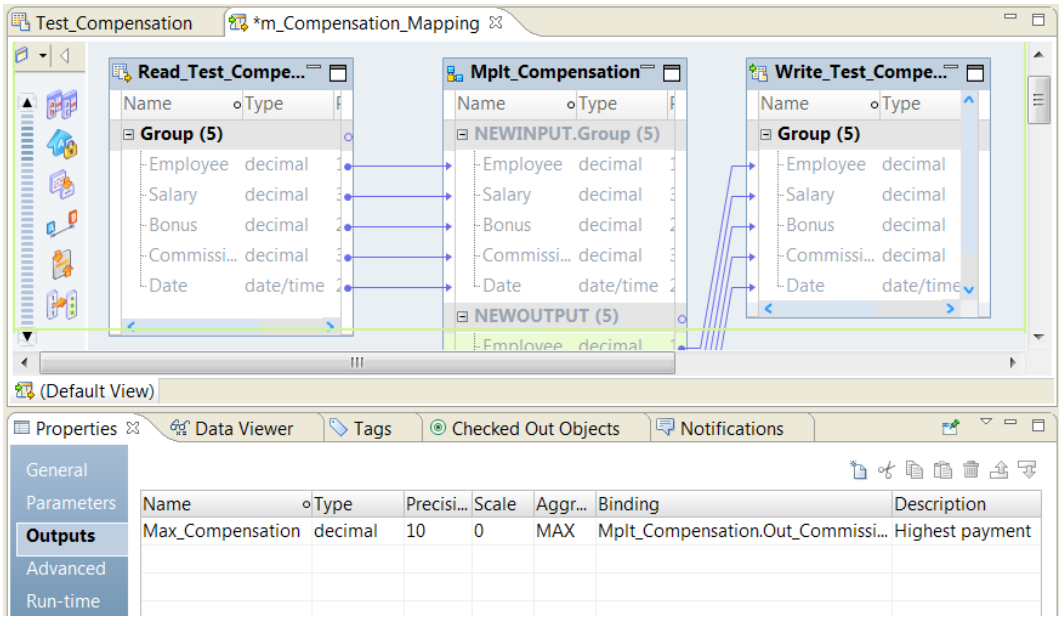
The following image shows the **Outputs Binding** dialog box:



After you select the outputs to bind to Max_Compensation, the **Binding** field contains the following text:

```
Mplt_Compensation.Out_Salary,Mplt_Compensation.Out_Bonus,Mplt_Compensation.Out_Commission
```

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.

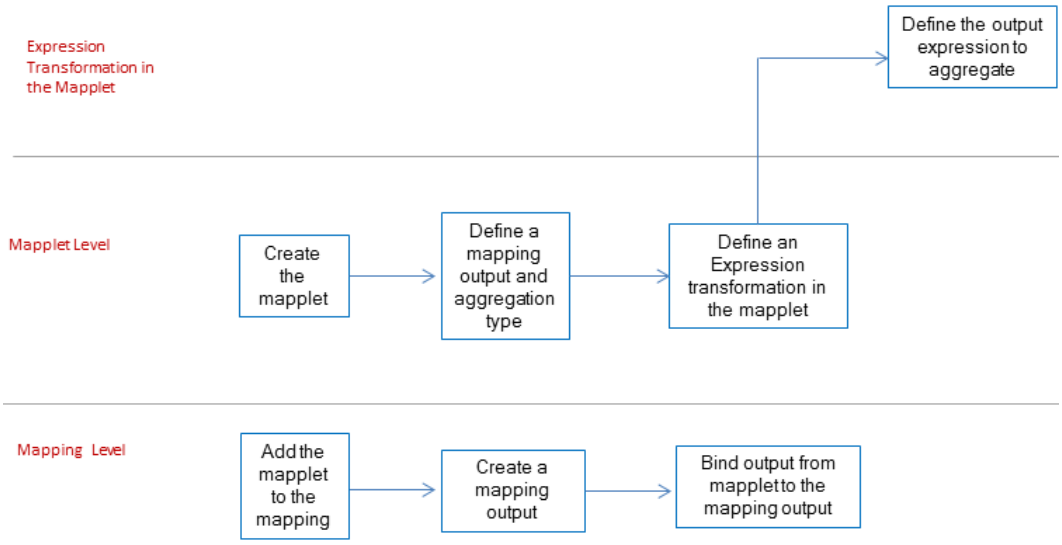
When you include the logical data object in a mapping, the Read or Write mapping computes the value of the mapping outputs. The logical data object passes the output values to the mapping. You can bind more than one output from the logical data object mapping to the same output at the mapping level.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

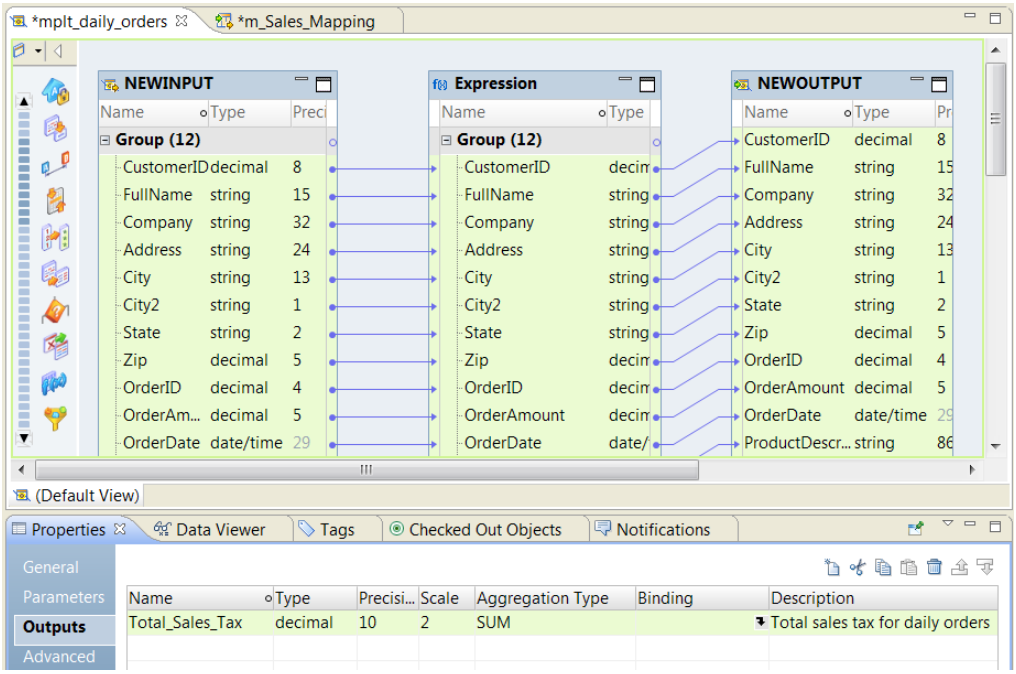
Defining Mapplet Outputs

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.

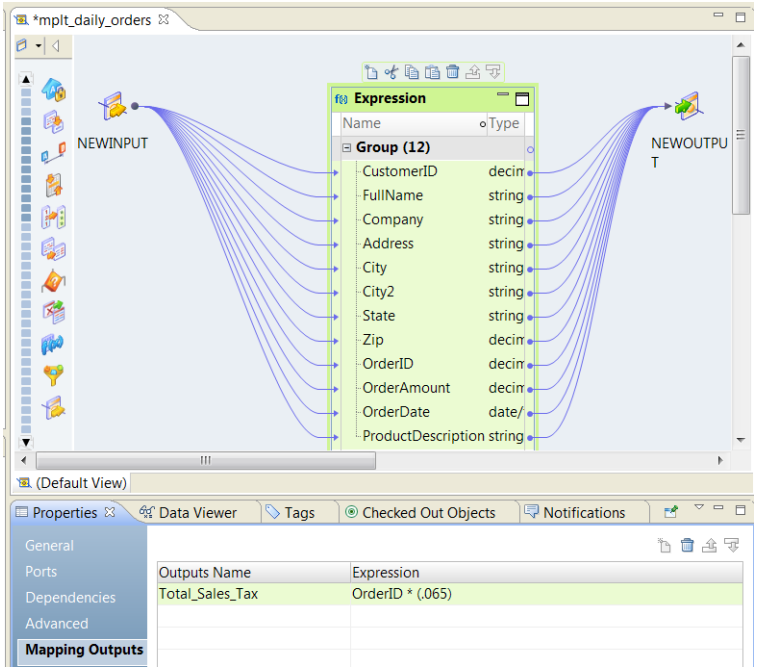
4. Enter an expression with the Expression Editor.

   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

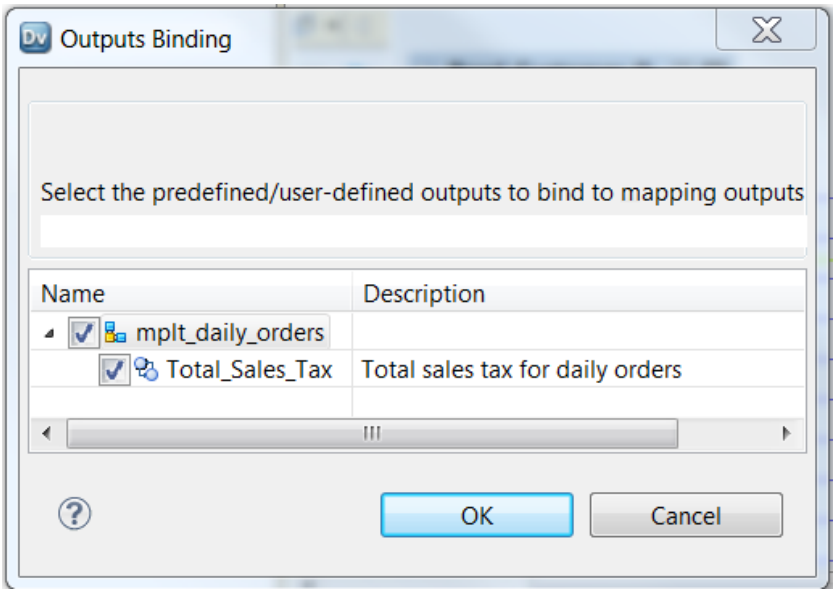
When you include the mapplet in a mapping, you can bind the outputs from the mapplet to the mapping outputs that you define at the mapping level.

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:

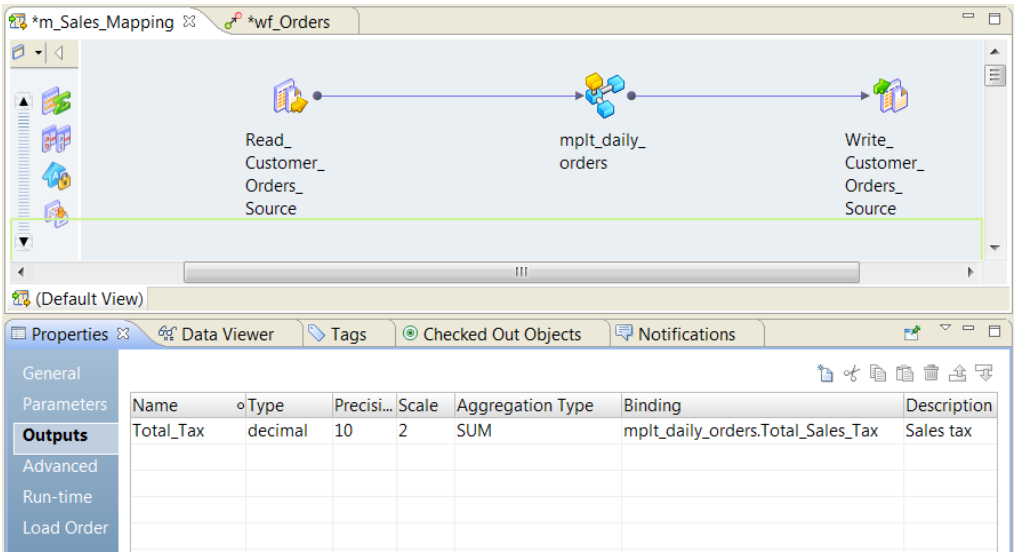


8. Select the mapplet ouput to bind to the mapping output.

   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:



CHAPTER 6

Generate a Mapping from an SQL Query

This chapter includes the following topics:

- Generate a Mapping from an SQL Query Overview, 117
- Example of Generated Mapping from an SQL Query, 117
- SQL Syntax to Generate a Mapping, 118
- Function Support in Queries that Generate a Mapping, 119
- Generating a Mapping or Logical Data Object from an SQL Query, 121
- Generate a Mapping from an SQL Statement, 121

Generate a Mapping from an SQL Query Overview

You can generate a mapping from an SQL query in the Developer tool. To generate a mapping, you can enter an SQL query or you can load a text file that contains the query. Optionally, you can define the source of the query table. The Developer tool validates the SQL query and generates a mapping.

You can also generate a logical data object from an SQL query that contains only SELECT statements.

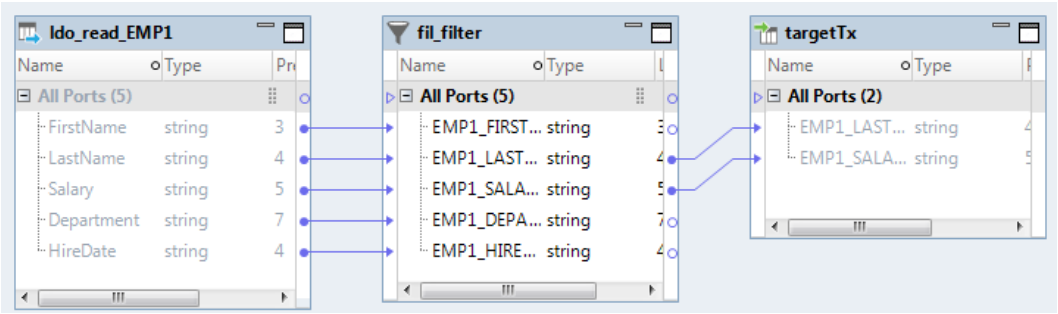
Example of Generated Mapping from an SQL Query

You have a table of employees, and you want a list of employee salaries for employees that were hired after January 1, 2001.

You create the following SQL statement:

```
SELECT LastName, Salary from emp1 where HireDate > 01/01/2001
```

The following image shows the mapping created from the SQL statement:



SQL Syntax to Generate a Mapping

You can use an ANSI-compliant SQL statement to generate a mapping in the Developer tool.

The Developer tool can generate a mapping from a standard SELECT query. For example:

```
SELECT column_list FROM table-name
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
```

If the SELECT SQL statement contains a correlated subquery, the query is valid if it is possible to flatten or rewrite the query as a single standard query.

ANSI SQL does not support some datatypes. For example, if the query requests results from a data source where one of the columns has the type timeStampTZ, the SQL is not valid.

Correlated Subqueries

A correlated subquery is a subquery that uses values from the outer query in its WHERE clause. The Data Integration Service flattens the correlated subqueries before it runs the query.

The following table shows the results of a correlated subquery that the Data Integration Service flattened:

Type	Query
Non-flattened	`SELECT huge.* FROM huge WHERE c1 IN (SELECT c1 FROM tiny)`
Flattened	`SELECT huge.* FROM huge, tiny WHERE huge.c1 = tiny.c1`

The Data Integration Service can flatten a correlated subquery when it meets the following requirements:

- The type is IN or a quantified comparison.
- It is not within an OR operator or part of a SELECT list.
- It does not contain the LIMIT keyword.
- It does not contain a GROUP BY clause, aggregates in a SELECT list, or an EXIST or NOT IN logical operator.
- It generates unique results. One column in the corelated subquery is a primary key. For example, if r_regionkey column is a primary key for the vs.nation virtual table, you can issue the following query:

```
SELECT * FROM vs.nation WHERE n_regionkey IN (SELECT b.r_regionkey FROM vs.region b WHERE
b.r_regionkey = n_regionkey).
```

- If it contains a FROM list, each table in the FROM list is a virtual table in the SQL data service.

Function Support in Queries that Generate a Mapping

Informatica supports functions that meet the ANSI SQL-92 standard.

In addition, some functions have specific syntax requirements.

The following table lists the functions and supported syntax:

Function	Syntax

DATE( )	To specify the format of a date:
	`DATE(format '<format>')`
	where <format> is a standard date format.
	Example:
	`SELECT  DATE(format 'dd-mm-yyyy') from table`

POSITION( )	To determine the position of a substring in a literal string:
	`POSTITION('<substring>', '<string>'`
	`)`
	Example:
	`POSITION('MA', 'James Martin')`
	To determine the position of a substring in a table column:
	`POSTITION('<substring>', <column_name>`
	`)`
	Example:
	`POSITION('MA', FULL_NAME)`

Generate a Mapping from an SQL Query with an Unsupported Function

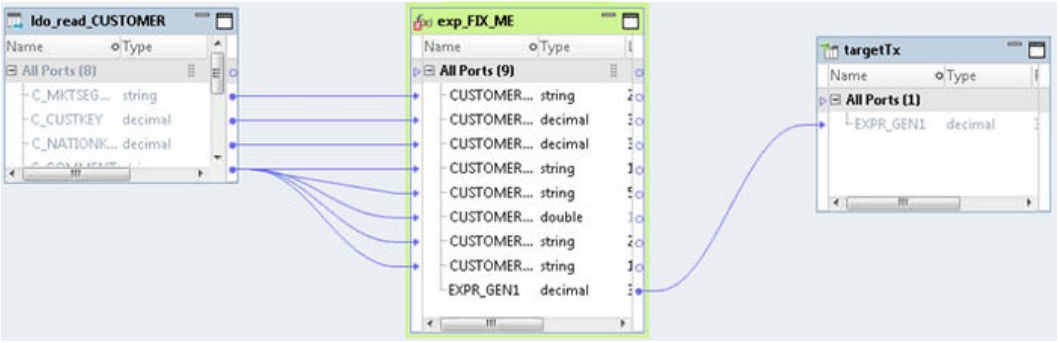
When the Developer tool generates a mapping from SQL, it validates the functions in the query. Use ANSI-compliant SQL to ensure valid mapping generation.

If the Developer tool encounters an unknown function in a valid SQL statement, it might generate a mapping that contains a transformation labeled FIX_ME or an expression labeled FIX_EXPR. Edit these objects to fix the mapping and get valid results. Unknown functions appear as a warning message in the mapping log file.

For example, you use the following SQL statement to generate a mapping:

```
SELECT unknownFunctionABC(c_custkey,c_comment) from customer
```

The following image shows how the mapping generated from this SQL statement includes an Expression transformation that requires fixing:



Notice that the Expression transformation is marked with an error icon. Use the Ports tab to edit the erroneous expression. The mapping is not valid until you correct the error.

INSERT, UPDATE and DELETE Syntax

Use the following syntax to create valid INSERT, UPDATE and DELETE statements:

- Use the following syntax for an INSERT statement:

```
INSERT INTO <TABLENAME> [<list>]
   <select query>
```

- Use the following syntax for an UPDATE statement:

```
UPDATE [schema .] { table | view} [ alias ]
   SET column = { expr | subquery }
   [, column = { expr | subquery }]... [WHERE condition]
```

- Use the following syntax for a DELETE statement:

```
DELETE FROM <Table> [[<AS>] <ALIAS>] [WHERE condition]
```

Rules and Guidelines for INSERT, UPDATE, and DELETE Statements

Consider the following rules and guidelines for INSERT, UPDATE, and DELETE statements:

- An INSERT, UPDATE, or DELETE statement creates source and target objects in the mapping that are logical data objects.
- Only one INSERT, UPDATE, or DELETE statement is valid. For example, a statement that contains an INSERT and a nested UPDATE statement is not valid.
- When the INSERT, UPDATE, or DELETE SQL statement contains a correlated subquery, the Developer tool cannot generate a mapping.
- An UPDATE or DELETE statement creates an Update Strategy transformation in a mapping. Because an Update Strategy transaction requires a primary key, the data target must contain a primary key. After mapping generation, verify the primary keys.
- The Developer tool ignores any INSERT statement in an ORDER BY clause because relational databases do not follow ordering when inserting data.

Generating a Mapping or Logical Data Object from an SQL Query

You can convert an SQL statement into a mapping or a logical data object. You might want to generate a logical data object to create an object that you can reuse in other mappings.

1. Click **File** > **New** > **Mapping from SQL Query**.
   The **Generate Mapping or Logical Data Object from SQL Query** dialog box opens.
2. Choose whether to enter an SQL query, or select a file that contains an SQL query.
   - To enter an editable SQL query, choose **Enter an SQL Query**, and then type or paste an SQL query in the editor. Then click **Validate Query**.
   - To select a file that contains an SQL query, choose **Select an SQL file**, and then browse to the file that contains an SQL query.

   The Developer tool validates the SQL syntax. If the syntax is not valid, you must fix it before you can proceed.
3. To generate a logical data object instead of a mapping, select **Generate Mapping as a Logical Data Object**.
4. Optionally, rename the mapping or logical data object that you want to generate.
5. Click **Next**.
   The dialog box displays the tables that correspond to data sources.
6. Click the row of the table under **Data Source** to select the data source for the mapping.
   If the table has a data source, you can click it to optionally change the data source.

   The **Select Data Source** dialog box opens and lists the tables on the Model repository that you can access.
7. Select any data source in the Model repository.
8. Click **Finish**.

The generated mapping or logical data object opens in an editor.

You can select any object in the mapping to view or edit it. Then you can run the mapping or include it in an application or workflow that you deploy to the Data Integration Service.

If you created a logical data object, you can reuse it in other mappings. For example, you can reuse the generated logical data object as the source in a mapping.

Generate a Mapping from an SQL Statement

To generate a mapping from an SQL statement, perform the following tasks:

1. Create an SQL statement.
2. Paste or import the SQL statement to the Developer tool, validate the SQL statement, and generate a mapping.
3. Complete mapping development. Follow these steps:
   a. Test and iteratively develop the mapping until it meets requirements.
   b. Deploy the mapping to the Data Integration Service.

Create an SQL Statement

To use an SQL statement to generate a mapping, create an SQL statement.

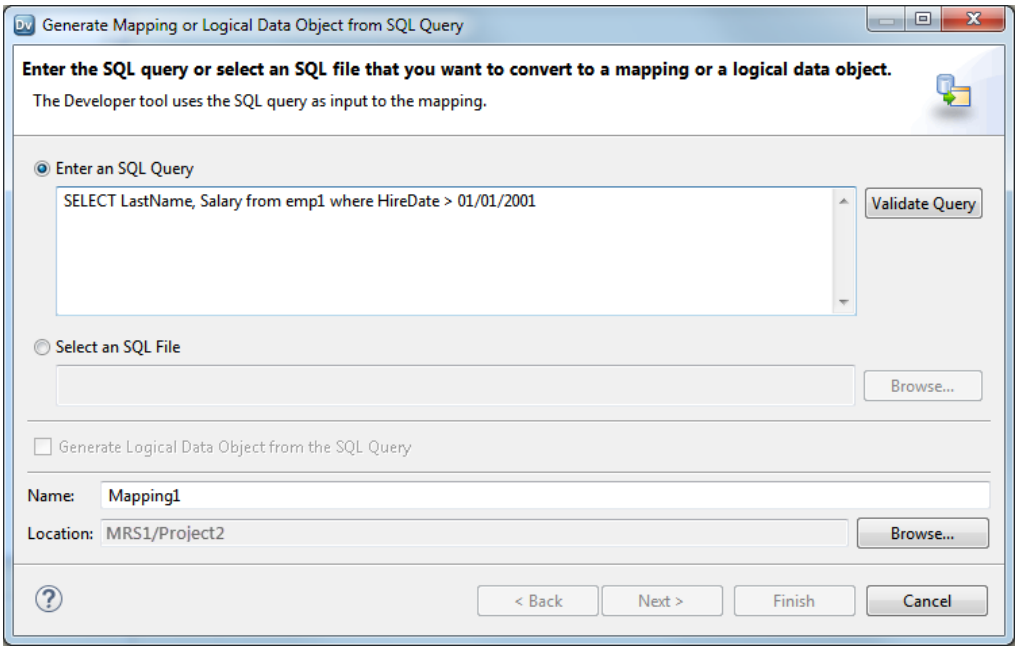
You can use an SQL query tool or write an SQL statement from scratch to create an SQL statement. Follow the syntax guidelines in this article.

Note: Some non-Informatica functions are supported. Others can be used in a valid query that generates a mapping with results that are not valid. For more information about function support in SQL statements, contact Informatica Global Customer Support.

Paste or Import the SQL Statement to the Developer Tool

1. Locate the SQL file that contains the SQL statement to import, or copy the entire statement to the clipboard.
2. In the Developer tool, click **File** > **New** > **Mapping from SQL Query**

   The **Generate Mapping or Logical Data Object from SQL Query** dialog box opens.



3. Import the query to the dialog box. Choose one of these methods:
   - Select **Enter an SQL Query** and paste the query from the clipboard to the editor.
   - Select **Select an SQL File** and browse to and select the file.
4. Click **Validate**.

   Developer tool validates the SQL statement. Correct any errors.
5. If you want to generate a logical data object instead of a mapping, select **Generate Logical Data Object from the SQL Query**.

   If you select this option, perform the following steps:

   1. Optionally rename the logical data object to create.
   2. Optionally click **Browse** to select a location for the Logical Data Object, or accept the default location.

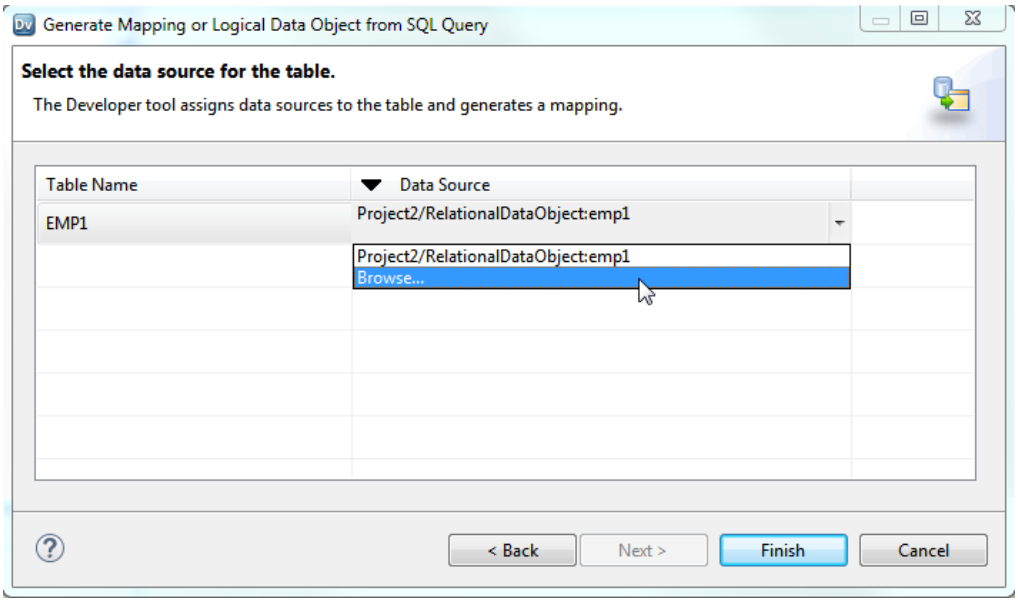
3. Click **Validate**.
   Developer tool validates the SQL statement. Correct any errors.

6. Click **Next**.

   The **Select the data source for the table** dialog box opens.

7. To select a data source for the table, click in the Data Source column, and then click **Browse**.

   The following image shows where to click **Browse** to select a data source:



8. Click **Finish**.

The Developer tool generates a mapping from the SQL query and opens the mapping in an editor.

Complete Mapping Development

After you create a mapping, perform the following steps to complete development of the mapping:

1. Run the mapping and view results.
2. Iteratively edit and re-run the mapping until it meets requirements.
3. Deploy and run the mapping on the Data Integration Service.
   You can deploy the mapping by itself, or include it in an application that you deploy. If you deploy the mapping by itself, the Data Integration Service creates an application to contain it.

   For more information about mappings, applications and deployment, see the *Informatica Developer Tool Guide*.

CHAPTER 7

Dynamic Mappings

This chapter includes the following topics:

- Dynamic Mappings Overview, 124
- Dynamic Mapping Configuration, 125
- Dynamic Sources, 128
- Dynamic Targets, 132
- Dynamic Ports and Generated Ports, 138
- Dynamic Expressions, 140
- Input Rules, 142
- Selection Rules and Port Selectors, 151
- Design-time Links , 153
- Run-time Links, 155
- Troubleshooting Dynamic Mappings, 158

Dynamic Mappings Overview

A dynamic mapping is a mapping that can accommodate changes to sources, targets, and transformation logic at run time. Use a dynamic mapping to manage frequent schema or metadata changes or to reuse the mapping logic for data sources with different schemas. Configure rules, parameters, and general transformation properties to create a dynamic mapping.

If a data source changes for a source, target, or lookup, you can configure a mapping to dynamically get metadata changes at run time. Configure parameters, rules, ports, and links within the mapping to receive and propagate changes at all stages of the mapping. You do not need to manually synchronize the data object and update each transformation before you run the mapping again. The Data Integration Service can dynamically determine transformation ports, transformation logic in the ports, and the port links within the mapping.

Dynamic Mapping Example

Every week, you receive customer data from different departments that you need to join and aggregate. The departments might periodically change the source schema to include additional columns for departmental analysis.

To accommodate the changes to the data source, you create a dynamic mapping. You configure the Read transformation to get data object columns at read time. Create an input rule to include columns that you need and to exclude all other columns.

Dynamic Mapping Configuration

If a source changes, you can configure the Read transformation to accommodate changes. For example, you can configure the transformation to use a different data source or to update the data object based on the data source. If a target changes, you can configure the Write transformation accommodate target changes. For example, you can configure the Write transformation to generate columns based on an associated data object or the mapping flow. If the target is relational, you can create or replace tables at run time.

Configure transformations in a mapping or mapplet to receive and propagate the changes throughout the mapping. Create dynamic ports to receive new or changed columns based on the data flow. A dynamic port generates a port for each incoming column. Configure input rules to determine the columns that a dynamic port receives and to rename or reorder the generated ports.

Create a dynamic expression by using dynamic ports or selection rules in the expressions. When you include a dynamic port, the expression runs against each port that the dynamic port generates. When you include a selection rule, the expression runs against each port in the rule.

When an Expression, Joiner, or Lookup transformation contains generated ports, you can configure port selection rules that accommodate changes to the generated ports when the mapping runs. For example, you need to perform a calculation on sales data, but the sales column name is different for each source. You create a rule to select the correct column to calculate.

You can use parameters to change values at run time. Use parameters to change values such as sources, targets, connections, and rules within the mapping.

Transformations might change in such a way that you cannot create direct links when you design the mapping. If you cannot create links at design time, you can configure run-time links. A run-time link uses a policy or parameter to determine the ports to link between transformation groups at run time.

Dynamic Data Sources

You can configure a mapping to accommodate changes to sources and targets at run time. A dynamic mapping can include flat file and relational data sources. You can use parameters and configure transformation properties based on the types of changes that you expect.

You can configure a mapping to accommodate run-time changes to the following data sources:

Sources

A dynamic source can include relational and flat file sources. Configure the Read transformation and the physical data object to accommodate run-time changes. You can change the source metadata based on the location of the file or source connection, changes to incoming source columns, or the data object.

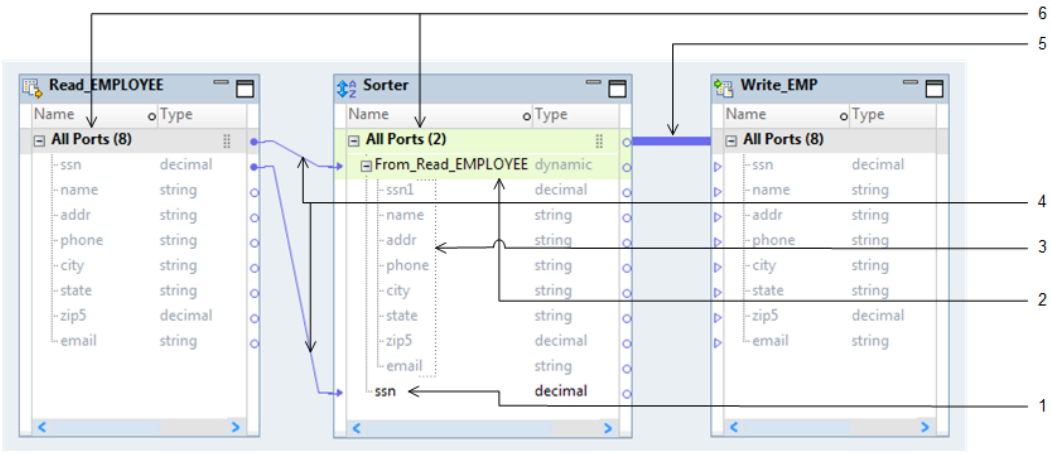
Targets

A dynamic target can include relational and flat file targets. You can define the target based on the mapping flow or the associated data object. You can also choose to get columns for the data object from the data source. You can use parameters to change run-time properties such as the target data object or the target connection.

Dynamic Mapping Ports and Links

In order to process changes in metadata, you can configure port and link types that a regular mapping does not contain.

The following image shows the ports and links that you can see in a dynamic mapping:



1. Static port (port)
2. Dynamic port
3. Generated port
4. Design-time link (link)
5. Run-time link
6. Port Group

Static port (port)

A port that you can create in any type of mapping, dynamic or non-dynamic. Data can pass in to and out of the port and does not contain any dynamic configuration.

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 creates a generated port for each column based on the dynamic port rules.

Design-time link (link)

A link that you create to connects ports that propagate data from one transformation to another. You also create these links in a regular mapping.

Run-time link

A link between transformation groups that the Data Integration Service uses to determine which ports to connect at run time based on a policy, a parameter, or both.

Port group

A set of ports in a mapping that represents a row of data. In a dynamic mapping, you can drag a group to a downstream transformation to create a dynamic port.

Dynamic Mapping Rules

Create rules within a dynamic transformation to control the ports that a dynamic port receives and the ports that it generates.

You can configure the following types of dynamic mapping rules:

Input rules

An input rule defines the ports that the dynamic port generates. You can choose to include or exclude ports. You can also choose to rename and reorder the generated ports.

Selection rules and port selectors

Create a selection rule to define the generated ports that the Data Integration Service processes at run time. Create selection rules within a port selector. A port selector contains ports that you can reference in an expression or in a join or lookup condition. You can configure more than one port selector in a transformation based on the metadata changes that you anticipate at run time.

Parameters in Dynamic Mappings

A parameter is a constant value that can change between mapping runs. Use parameters in a dynamic mapping to change sources and targets for flat files or relational resources. You can also use parameters to change the input rules, selection rules, transformation properties, and run-time links.

The following table lists the functionality of parameters that you can create for the dynamic mapping components:

Dynamic Mapping Component	Parameter Functionality
Aggregator transformation	Change the group by port.
Joiner transformation	Change the join condition.
Lookup transformation	Change the lookup condition.
Rank transformation	Change the group by port.
Read transformation	Create parameters to perform the following tasks: - Change the input file name or directory of a flat file source. - Change the connection of a relational source. - Change a flat file data object, customized data object, or a relational data object.
Rules	Create parameters to perform the following tasks: - Change the input rule criteria by name or pattern. - Change the selection rule criteria by name or pattern.
Run-time links	Change the set of ports to link between transformation groups.
Sorter transformation	Change the sort key.
Write transformation	Create parameters to perform the following tasks: - Change the output file name or directory of a flat file target. - Change the connection of a relational target. - Change a flat file data object, customized data object, or a relational data object.

Related Topics:

- "Mapping Parameters Overview" on page 49

Dynamic Sources

A dynamic source is a source that can change at run time. You can configure flat file, relational, Amazon S3, Amazon Redshift, HBase, HDFS, Microsoft Azure Blob Storage, Microsoft Azure Data Lake Store and complex file dynamic sources in a mapping.

You can configure dynamic run-time functionality for a source in the following ways:

Get columns from the data source.

When you expect small changes to a source at run time, you can configure the Read transformation to get flat file or relational object columns at run time. 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.

Assign a parameter to determine the flat file name and directory of source.

When the flat file sources are similar, you can assign a parameter to a file name or directory. When you use a parameter, you do not need to create a data object for each source.

Assign a parameter to determine the resource, table owner, or directory of a relational data object.

When the relational sources are similar, you can assign a parameter to get the resource, connection, and table owner properties.

Assign a parameter to determine the data object to use for a file or relational source.

When you expect small changes to a 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.

The following table shows where you can configure the dynamic run-time functionality of a source:

Dynamic Run-time Source Functionality	Configuration
Get columns from the data source.	Configure the **Data Object** tab on the Read transformation for the following source types: - Flat file - Relational - Amazon Redshift - Amazon S3 - Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store - HBase - HDFS - Snowflake
Assign a parameter to determine the flat file name and directory.	Configure the **Advanced** tab on the physical data object for the following source types: - Flat file - Amazon Redshift - Amazon S3 - Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store - HDFS - HBase - Snowflake
Assign a parameter to determine the connection, owner, or resource.	Configure the **Run-Time** tab on the Read transformation for the following source types: - Relational - Amazon Redshift - Amazon S3 - Snowflake
Assign a parameter to determine the data object.	Configure the **Data Object** tab on the Read transformation for the following source types: - Flat file - Relational - Amazon Redshift - Amazon S3 - HDFS - Snowflake

Get Columns from the Data Source

You can update the ports in a Read transformation at run time based on the structure of the relational or flat file data source. You might use this method to update the run-time instance of the Read transformation.

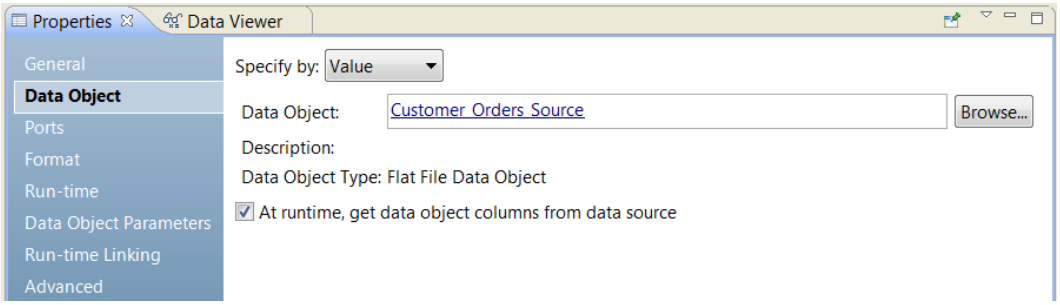
Update the columns at run time when you expect small changes to the source. For example, you need to process a source from another organization, but the organization cannot guarantee the order of the columns in the source file. When you enable the option to update the data object columns at run time, the Data Integration Service changes the ports of the Read transformation based on the structure of the source data. The Read transformation passes the data to the downstream transformations in the dynamic mapping for processing.

When you update the data object columns at run time, the Data Integration Service updates a run-time instance of the Read transformation. It does not update metadata in the Model repository. To see the

changes in the Developer tool, view the mapping with resolved parameters. To update the physical data object definition in the Model repository, use the synchronize option in the Developer tool. The Developer tool re-imports the physical data object metadata and changes the metadata.

Note: You cannot use a custom SQL query in a dynamic mapping.

The following image shows where to enable the option on the **Data Object** tab:



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 determines the structure of a flat file source based on the way you configure the flat file physical data object. You configure the data object to generate column names at run time.

Configure this functionality on the **Data Object** tab of the Read transformation for a flat file or relational source.

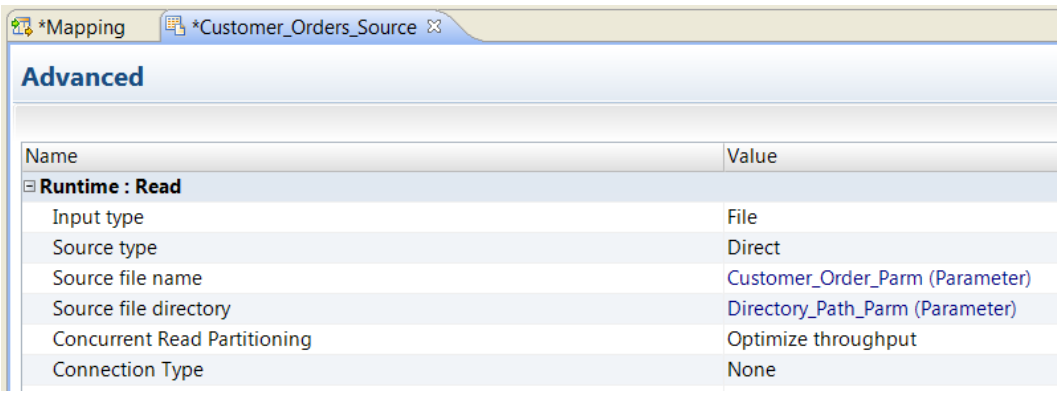
For more information about setting the flat file physical data object properties, see the *Informatica Developer Tool Guide*.

Assign a Parameter to a Flat File Name

To run a dynamic mapping with similar flat file sources, you can assign a parameter to a file name or directory. When you use a parameter, you do not need to create a data object for each 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:



Configure this functionality on the **Advanced** tab of the physical data object for a flat file source.

Assign a Parameter to Relational Source Properties

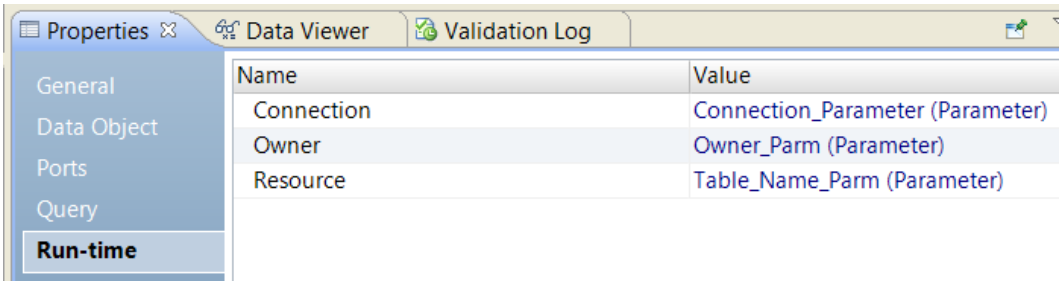
To run a dynamic mapping with similar relational sources, you can assign a parameter the resource, connection, and table owner properties in the Read transformation.

Use a parameter for the resource to run a mapping with different but similar tables in the same database. When you use a parameter for the resource, you do not need to create a data object for each source. Use a parameter for the connection to access a different database. You might need to run a unique SQL query against multiple relational sources.

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. Once you configure the run-time parameters for the Read transformation, the data source that the Read transformation uses is updated based on the data stored in the relational database.

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:



Configure this functionality on the **Run-Time** tab of the Read transformation for a relational source.

Assign a Parameter to the Source Data Object

You can assign a parameter to the data object and change the source for the Read transformation at run time.

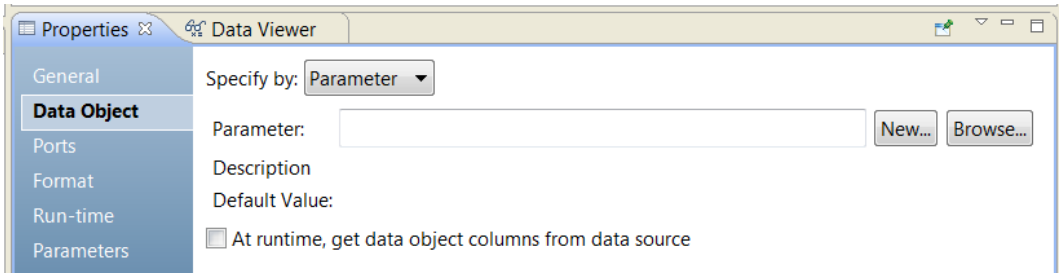
Parameterize the data object when you have a different physical data object in the Model repository for each data source. You might parameterize the data object if you need to configure the same transformation for a flat file or for a database table. When you parameterize the data object, you enable 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 the data object, create a resource type parameter or browse for a resource parameter. 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 following table describes the parameter options on the **Data Object** tab:

Parameter Options	Description
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.

Configure this functionality on the **Data Object** tab of the Read transformation for a flat file or relational source.

Dynamic Targets

A dynamic target is a target that can change at run time.

When you run a mapping, a dynamic target can get metadata changes from physical data targets, including relational tables, flat files, Amazon Redshift, Amazon S3, Microsoft Azure Data Lake Store, Microsoft Azure Blob Storage, HDFS, HBase and customized data objects. It can also generate columns based on the upstream column definitions.

You can configure dynamic run-time functionality for a target in the following ways:

Get columns from the data source.

When you expect small changes to the target, you can configure the Write transformation to get relational object columns at run time. When you configure the Write transformation to get metadata from targets, you can configure the Write transformation to update dynamically and remain synchronized with target objects.

Define target columns based on the mapping flow.

When you define columns based on the mapping flow, target columns are determined by upstream transformations.

Define target columns based on the data object.

When you define columns based on the data object, target columns are determined by the associated data object.

Define a target schema strategy.

You can define a target schema strategy to retain the target schema, or to create or replace target tables at run time. You can also specify a parameter value for the target schema strategy in the Write transformation.

When you configure a target schema strategy in the Write transformation to create or replace the target at run time, the Data Integration Service creates the target based on the data object or on the mapping flow. You can also define a DDL query to create the target based on the query.

Assign a parameter to determine the resource, table owner, or directory of a relational data object.

When the relational targets are similar, you can assign a parameter to get the resource, connection, and table owner properties.

Assign a parameter to determine the data object to use for a file or relational target.

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

The following table shows where you can configure the dynamic run-time functionality of a target:

Dynamic Run-time Target Functionality	Configuration
Get columns from the data source.	Configure the **Data Object** tab on the Write transformation for the following target types: - Relational - Amazon Redshift - Amazon S3 - Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store - HBase - HDFS - Snowflake
Define target columns based on the data object or the mapping flow.	Configure the **Ports** tab on the Write transformation for the following target types: - Flat file - Relational - Amazon Redshift - Amazon S3 - Microsoft Azure Blob Storage - Microsoft Azure Data Lake Store - HBase - HDFS - Snowflake
Select a target schema strategy.	Configure the **Advanced** tab on the physical data object for the following target types: - Relational - Amazon Redshift - Snowflake

Dynamic Run-time Target Functionality	Configuration
Define a DDL query to create the target table at run time.	Configure the **Advanced** tab on the physical data object for the following target type: - Relational
Assign a parameter to determine the connection, owner, or resource.	Configure the **Run-Time** tab on the Write transformation for the following target types: - Relational - Amazon Redshift - Amazon S3 - Snowflake
Assign a parameter to determine the data object.	Configure the **Data Object** tab on the Write transformation for the following target types: - Flat file - Relational - Amazon Redshift - Amazon S3 - Snowflake

Get Columns from the Data Source

You can update the ports in a Write transformation at run time based on the structure of the relational data source.

Update the columns at run time if you expect small changes to the target columns. When you get the data object columns from the data source at run time, the Data Integration Service creates a run-time instance of the data object based on the structure of the target. It does not update metadata in the Model repository. To view the run-time instance of the data object in the Developer tool, view the mapping with resolved parameters.

Note: If you configure the Write transformation to get columns form the data source and to define targets based on mapping flow, the mapping fails.

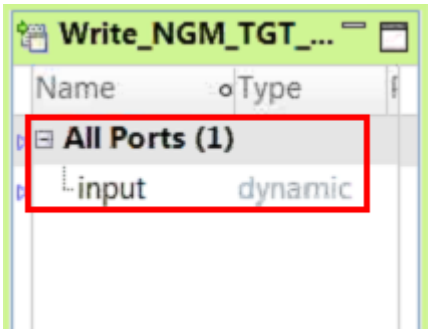
Configure this functionality on the **Data Object** tab of the Write transformation for relational targets.

Define Targets Based on the Mapping Flow

When you define columns by mapping flow, target columns are determined by upstream transformations. When an upstream transformation changes the port order and metadata, the Write transformation picks up the changes.

You can retain keys when you create or replace the target if the key columns in the upstream transformation match the key column names in the target.

The following image shows how the Write transformation appears when you define target columns based on mapping flow:



Note: To avoid unexpected results, do not configure a run-time link to a Write transformation that defines targets based on the mapping flow.

Configure this functionality on the **Ports** tab of the Write transformation for flat file and relational targets.

Define Targets Based on the Data Object

You can configure the Write transformation to define target columns based on the associated data object.

When you define target columns based on the data object, the Write transformation contains dynamic and generated ports.

You can choose to create or replace the target at run time. You can retain target keys when you create or replace the target if the column names match. You can configure rules to ensure that column names match.

Configure this functionality on the **Ports** tab of the Write transformation for flat file and relational targets.

Define a Target Schema Strategy

You can define a target schema strategy for the target table in the Write transformation.

You can select from the following target schema strategy options:

RETAIN - Retain existing target schema

The Data Integration Service retains the existing target schema.

CREATE - Create or replace table at run time

The Data Integration Service drops any existing target table associated with the write object and creates a table based on the data object or the mapping flow.

When the Data Integration Service creates a table based on the data object, the table contains columns that match the ports in the data object. If you create or replace the target at run time with a customized data object, the Data Integration Service creates a table with the name referenced in the data object connection.

When the Data Integration Service creates a table based on the mapping flow, the table contains columns that match generated ports in the Write transformation.

Assign Parameter

You can assign a parameter to represent the value for the target schema strategy and then change the parameter at run time.

Configure this functionality on the **Advanced** tab of the data object.

Define a DDL Query

At run time, the Data Integration Service can create or replace relational and Hive target tables based on a DDL query that you define.

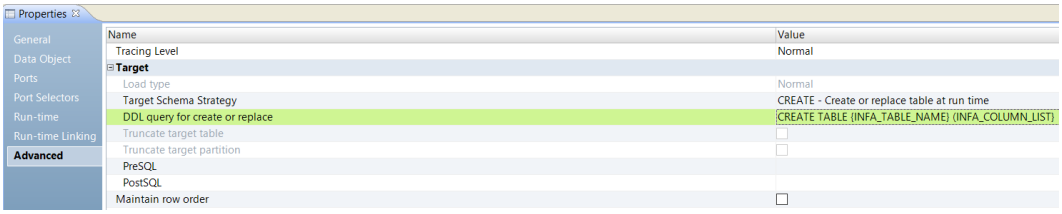
When you configure a target schema strategy in a Write transformation to create or replace the target table, by default, the Data Integration Service drops any existing target table associated with the write object. The Data Integration Service then creates a table based on the mapping flow or data object.

If you want to customize the table or specify additional parameters such as partitions, you can define a DDL query based on which the Data Integration Service must create or replace the target table. The table contains the columns that you define in the DDL query.

To create a partitioned Hive table, use the following syntax:

```
CREATE TABLE `hiveTable` (`Field` STRING) PARTITIONED BY ({INFA_PORT_SELECTOR:
PortSelector})
```

The following image shows the **DDL query** field:



You can enter placeholders in the DDL query. The Data Integration Service substitutes the placeholders with the actual values at run time. For example, if a table contains 50 columns, instead of entering all the column names in the DDL query, you can enter a placeholder.

You can enter the following placeholders in the DDL query:

INFA_TABLE_NAME

Fetches the target table name at run time.

INFA_COLUMN_LIST

Fetches a list of columns in the target table at run time.

INFA_PORT_SELECTOR

Adds a port selector.

You must enclose the placeholders within two curly brackets. For example, {INFA_TABLE_NAME}.

Configure this functionality on the **Advanced** tab of the data object.

Rules and Guidelines for Creating or Replacing the Target at Run Time

Consider the following rules and guidelines when you create or replace the target at run time:

- If the target table has a cyclic dependency among other tables in the database, the database cannot execute the command to drop or create the table, and the mapping fails.
- When the Data Integration Service replaces a target, it does not retain the indexes and permissions for the target table.
- If you do not configure the Write transormation to have dynamic ports, the Data Integration Service creates the target with linked and unlinked ports based on the data object. It writes data to the linked ports.

- The Data Integration Service creates a table even if the the resource in the data object is a synonym or view. While each connection can point to different database instances, all the connections in a dynamic mapping must all be of the same database type.

Assign a Parameter to Relational Target Properties

To run a dynamic mapping with similar relational targets, you can assign a parameter the resource, connection, and table owner properties in the Write transformation.

Use a parameter for the resource to run a mapping with different but similar tables in the same database. When you use a parameter for the resource, you do not need to create a data object for each target. Use a parameter for the connection to access a different database.

You cannot parameterize these properties in the physical data object. When you create parameters for the properties in the Write 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.

Configure this functionality on the **Run-Time** tab of the Write transformation for a relational target.

Assign a Parameter to the Target Data Object

You can assign a parameter to a customized data object and change the source for the Write transformation at run time.

Parameterize the data object when you have a customized data object in the Model repository for more than one target data source. When you change the value of the parameter, the target changes for all objects that uses the parameter.

When you create a Write transformation from a customized 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 definition from the Model repository. To parameterize the data object, create a resource type parameter or browse for a resource parameter. The parameter default value is the name of customized data object in the Model repository. When you create a default parameter value, you select a customized 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 table describes the parameter options on the **Data Object** tab:

Parameter Options	Description
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.

Parameter Options	Description
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 customized data object name. Read-only.

Configure this functionality on the **Data Object** tab of the Write transformation for a relational target.

Rules and Guidelines for Dynamic Targets

Consider the following rules and guidelines when you work with dynamic targets:

- When you preview a dynamic target, the Developer tool does not refresh the schema definition. If there is a mismatch due to changes in the schema such as a configuration to get columns from the data source or to replace the target at run time, the data preview fails. Manually synchronize the Read or Write transformation. If you continue to get an error, run the mapping to see the results.
- If the dynamic target is too small for the incoming data, the mapping fails with a message indicated that the value is too large for the column.
- 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.

Dynamic Ports and Generated Ports

You can create dynamic ports in a transformation to receive new or changed columns from an upstream transformation. A dynamic port receives one or more columns and generates ports based on input rules. The input rules determine the columns that a dynamic port receives and generates.

Use dynamic ports to perform the following tasks:

Receive new and changed columns.

To get data from a dynamic source or a parameterized source, create a dynamic port in the downstream transformations to receive new and changed columns. If a mapping contains from a dynamic source, the dynamic ports in the downstream transformations automatically get 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.

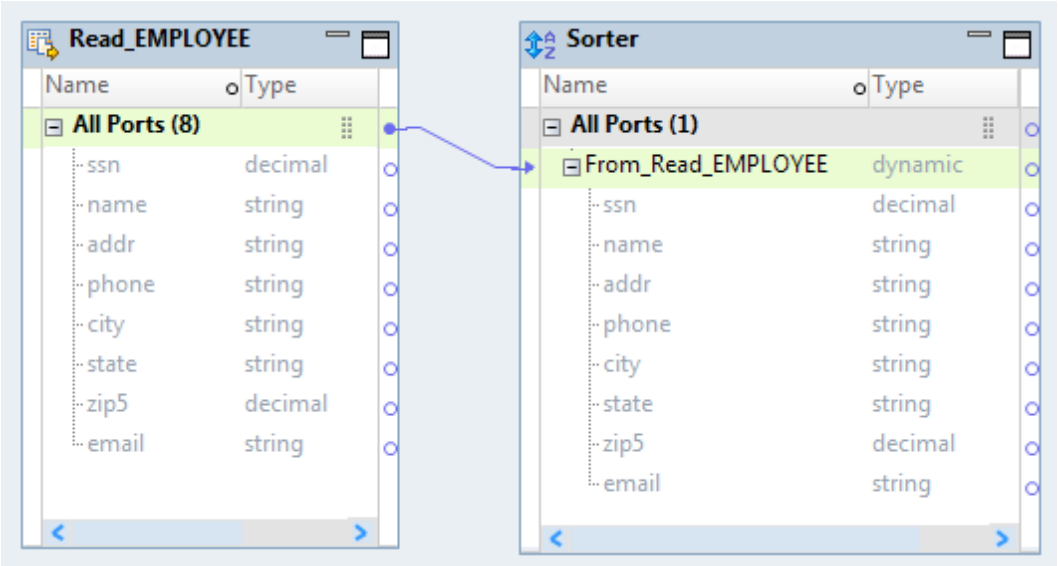
Filter columns based on input rules.

To process only a certain type of column 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 types. 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.

Repeat a calculation for more than one port in an Expression transformation

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 image shows a dynamic port named From_Read_Employee and the generated ports:



Dynamic and Generated Port Configuration

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 create more than one dynamic port in a transformation.

When you use the **New** button to create a port, the Developer tool assigns a default name. Rename the dynamic ports to ensure that the port names within each transformation are unique. 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.

You can create dynamic ports in the following transformations:

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

If the mapping contains transformations that cannot include dynamic ports, you might need to manually update the mapping when the source metadata changes.

Note: Any change to the port attributes propagates to the generated ports in the pipeline. You do not need to manually propagate the changed port attributes.

Rules and Guidelines for Dynamic and Generated Ports

Consider the following rules and guidelines when you work with dynamic and generated ports:

- You cannot link a generated port to an Output transformation in a virtual table mapping.
- You cannot link a generated port to a Fault, Input, or Output transformation in an operation mapping.

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:

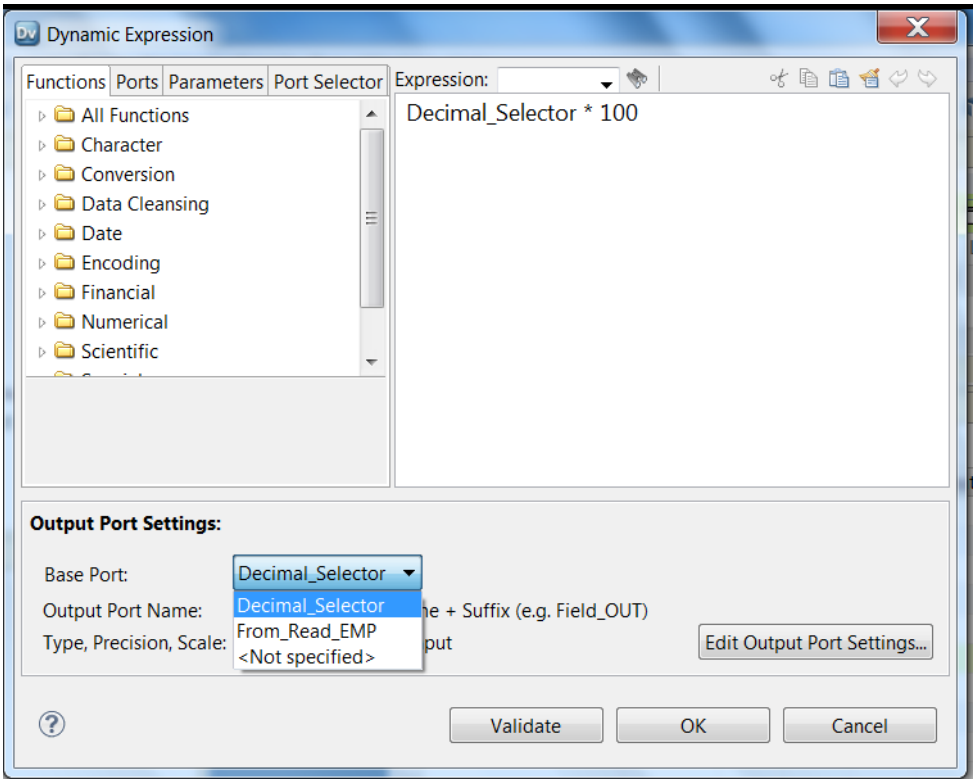
```
EMPNO   Decimal
NAME    String
SALARY  Decimal
DEPTNO  Decimal
```

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.

Parameterizing the Dynamic Expression

When you use a dynamic expression in an Aggregator, Expression, or Rank transformation, you can create an expression parameter. The expression parameter contains a complete expression.

Example

You work for a consulting firm in the infrastructure sector that collects data on the forces that weather phenomena exert on different locations in the architecture of a building. You want to use a dynamic mapping to transform the data that you collect and round the value of the force that is exerted.

In the dynamic mapping, you use an Expression transformation that receives the following generated input ports:

```
Force_Location_A   Decimal
Force_Location_B   Decimal
Force_Location_C   Decimal
```

You create a port selector `Port_Selector_Force` that has a selection rule to include ports that have the prefix `Force_`. You also create a dynamic output port `Dynamic_Forces` and use an expression parameter for the dynamic expression. The default value of the expression parameter is the following expression:

```
CEIL(Port_Selector_Force)
```

You can also use the expression parameter in multiple parameter sets. For example, the default value of the expression parameter in the one parameter set might be `CEIL(Port_Selector_Force)`. In another parameter set, the default value of the expression parameter might be `FLOOR(Port_Selector_Force)`.

Input Rules

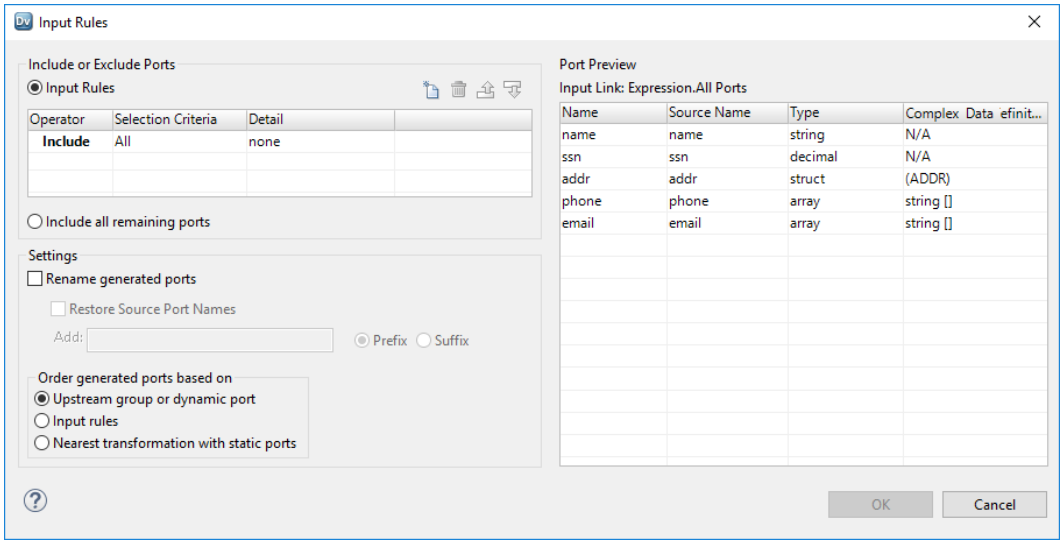
Input rules are a set of conditions in a dynamic port that define the ports to generate. You can configure input rules to filter the columns that a dynamic port receives and to create generated ports for the filtered columns. Use input rules to propagate specific columns downstream in the mapping.

To process certain columns or certain type of columns in a dynamic port, define input rules to filter the columns based on a name, data type, pattern, or complex data type definition. For example, an employee database table has 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.

Input Rule Configuration

Use the **Input Rules** dialog box to define the ports to include, to rename the generated ports, to 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:



When you configure input rules, you configure the following properties:

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

Include all remaining ports

Add ports that are excluded from other dynamic ports in the transformation. When a transformation contains multiple dynamic ports, 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 the transformation where ports are generated or to ensure that the port names are unique within each transformation.

Restore source port names

Restore the source port names in the generated ports. The Developer tool restores the port names as they appear in the nearest upstream transformation with static ports.

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 that they appear in the upstream transformation. You can also change the order according to the order of ports in the Read transformation. However, if one or more mid-stream transformations have dynamic ports and static ports, the Developer tool displays the ports in the order that they appear in the nearest upstream transformation with static ports.

After you configure the rules, you can preview the generated ports to verify the combination of rules. The Data Integration Service evaluates rules in the order that they appear in the **Input Rules** dialog box. You can change the order of the rules to ensure that they run in the correct order.

Include or Exclude Ports

You can include or exclude ports based on the port name 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. When you choose the selection criteria, an input rule detail dialog box appears based on the criteria. For example, you provide the **Name** selection criteria details in the **Input Rule Detail By Name List** dialog box

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:

Selection Criteria	Description	Criteria Details
All	Includes all ports. Do not use this selection criteria with exclude operator.	You do not have to specify any details.
Name	Filters ports based on the port name.	Select the port names from a list of values or use a parameter of type Port or Port List. You can also select the port names by source name. **Note:** Name values are not case sensitive.
Type	Filters ports based on the data type of the port.	Select data types from a list.

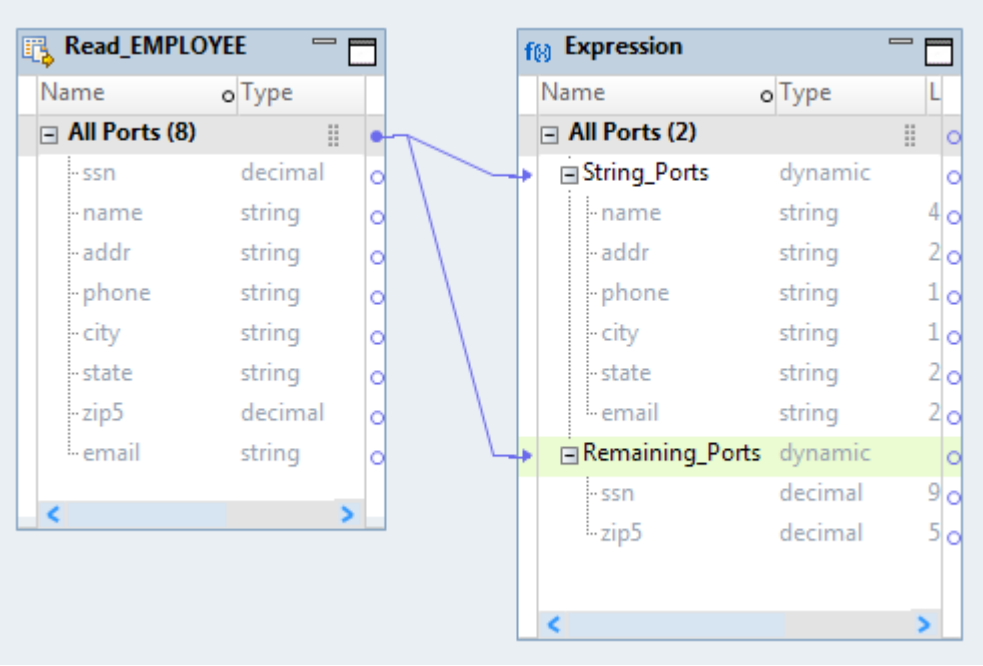
Selection Criteria	Description	Criteria Details
Pattern	Filters ports based on the pattern of the port name.	Choose prefix, suffix, or regular expression as the pattern type for the port name. Then, enter a value for the pattern or use a parameter of type String. You can also select the ports by source name to find the pattern in the source name. **Note:** Pattern values are not case sensitive.
Complex Data Type Definition	Filters ports based on the complex data type definition.	Choose prefix, suffix, or regular expression as the pattern type for the complex data type definition. Then, enter a value for the pattern or use a parameter of type String. **Note:** Pattern values are not case sensitive.

Include All Remaining Ports

When a transformation contains more than one dynamic port, you can configure the last dynamic port to include all ports that were not included in any other dynamic port.

For example, you want to 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. In an Expression transformation, you create two dynamic ports. You configure input rules to include all string data in one port and to put all remaining date in the other port. You choose the **Include all remaining ports** option for the last dynamic port.

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

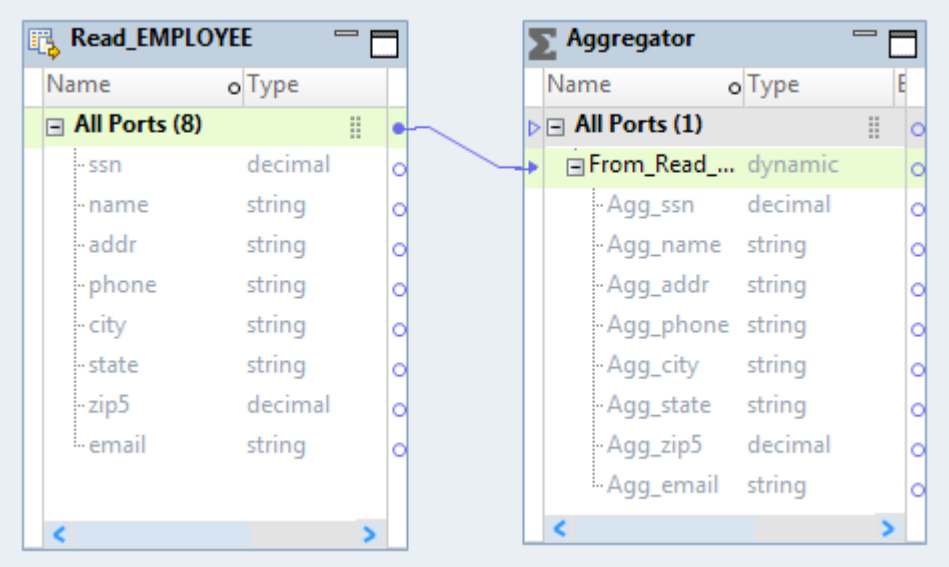


Rename Generated Ports

You can generate unique port names through a setting that renames generated ports with a prefix or suffix.

For example, you can add an `Agg_` prefix to indicate that the ports were generated in an Aggregator transformation.

The following image shows the renamed generated ports in the Aggregator transformation with an `Agg_` prefix:



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. You might want to rename generated ports if the Data Integration Service will not be able to resolve port conflicts at run time. If the mapping uses a dynamic source, the Data Integration Service might encounter a port name conflict at run time. If the Data Integration Service encounters a port name conflict, it tries to rename the generated port. The mapping fails if the Data Integration Service cannot resolve the port name conflict. The mapping fails in the following situations:

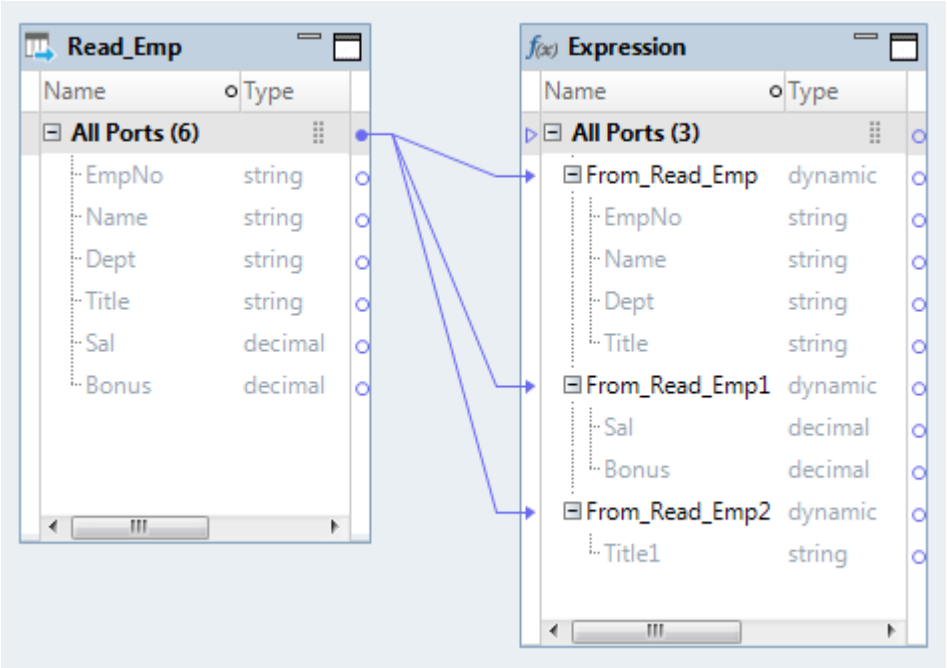
- An unresolved link exists from the renamed generated port to a static port.
- A transformation property, such as group by port or join condition, uses the renamed generated port.

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

Example - Rename Generated Ports

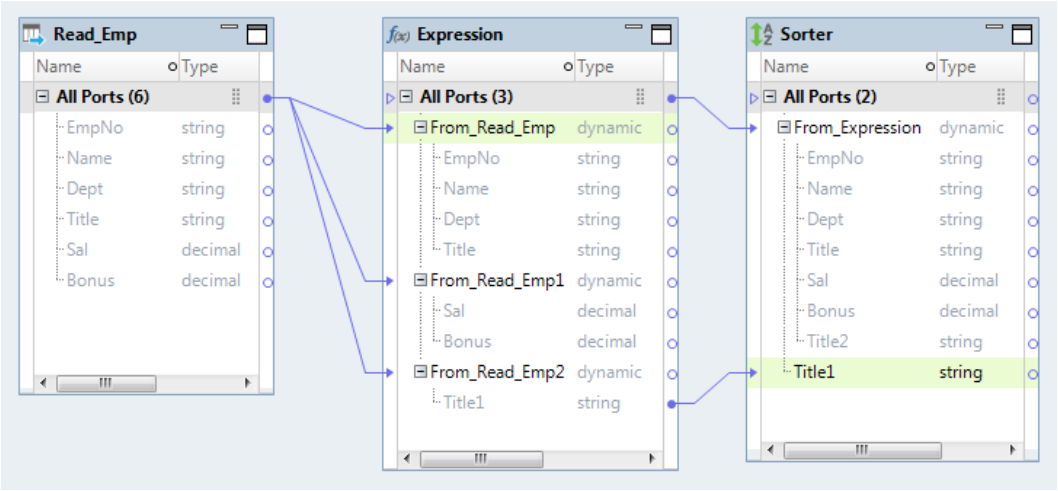
An Expression transformation has three dynamic ports. The "From_Read_Emp" and "From_Read_Emp2" dynamic ports include the generated port "Title." To avoid a name conflict, the Developer tool renames the generated port in "From_Read_Emp2" to "Title1."

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



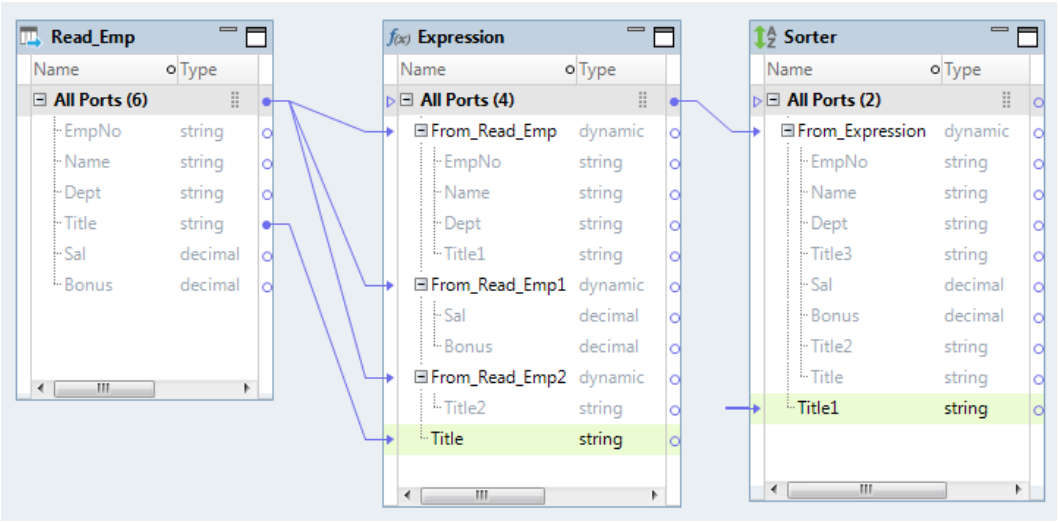
You link the generated port "Title1" in the Expression transformation to a port "Title1" in the Sorter transformation. You also use "Title1" as the sort key.

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.

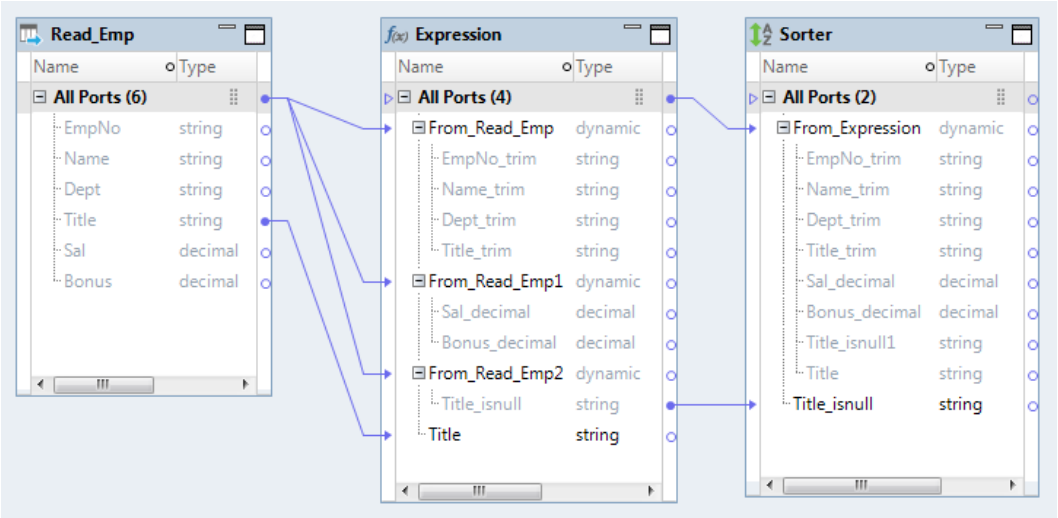
The following image shows the new link between the Read and Expression transformations, the generated ports that the Developer tool renamed in the Expression transformation, and the unresolved link to the Sorter transformation:



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 mapping failure, rename generated ports to ensure 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:



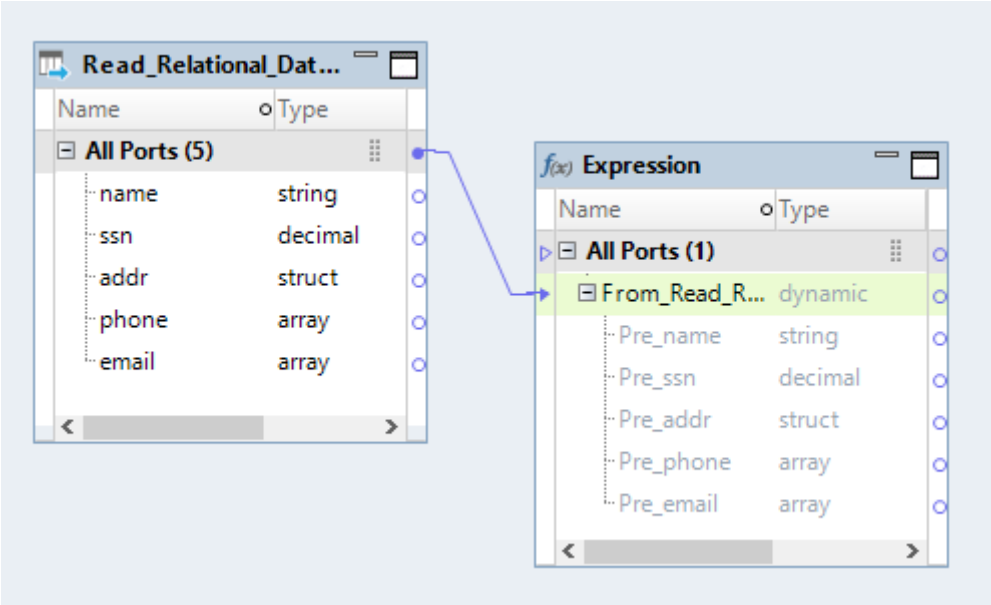
Restore Source Port Names

You can restore source port names when you rename generated ports.

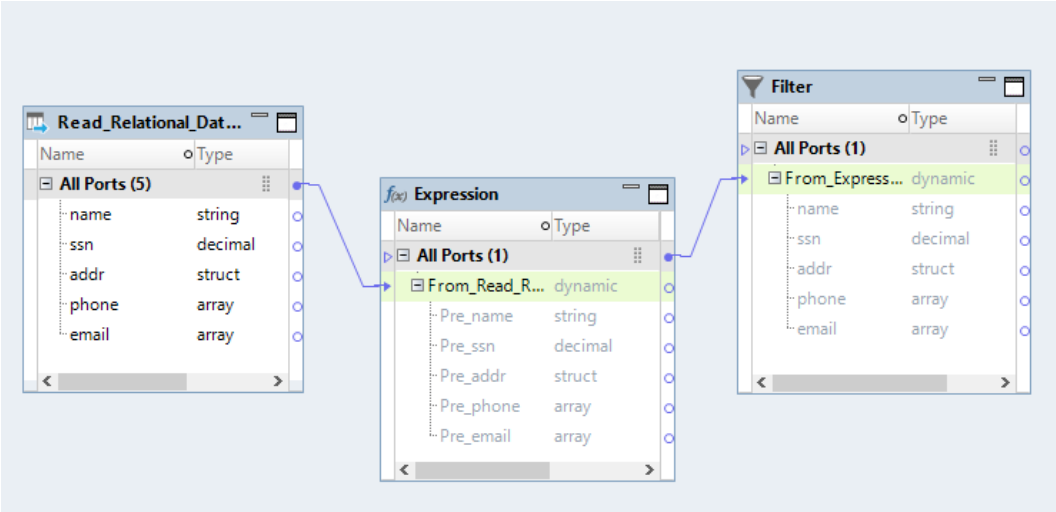
For example, you can rename generated ports to add a prefix `Pre_` to all of the ports in one transformation in the mapping flow. In a downstream transformation, you can restore the source port names to restore the

name before the prefix was added. When you restore the source port names, the Developer tool restores the port names as they appear in the nearest upstream transformation with static ports.

The following image shows an Expression transformation where the generated ports are renamed with a prefix `Pre_`:



The following image shows a Filter transformation where the source port names are restored:



The port names are restored according to the upstream Read transformation that contains static ports.

Reorder Generated Ports

You can reorder generated ports through a setting that reorders the ports based on the order of input rules or the ports in the Read transformation. By default, the Developer tool displays the generated ports in the same order that they appear in the upstream transformation.

You can choose one of the following options to reorder generated ports:

Upstream group or dynamic port

Display the ports in the same order that they appear in the group or the dynamic port of the upstream transformation. This is the default option.

Input rules

Display the generated ports based on the order of the input rules for the dynamic port.

The Data Integration Service reads rules in the order listed in the **Input Rules** dialog box. Review the port order and reorder them based on the order of the input rules. You can ensure that the Data Integration Service processes ports and rules in the order that you require. Reordering the ports also helps you view and analyze results.

Nearest transformation with static ports

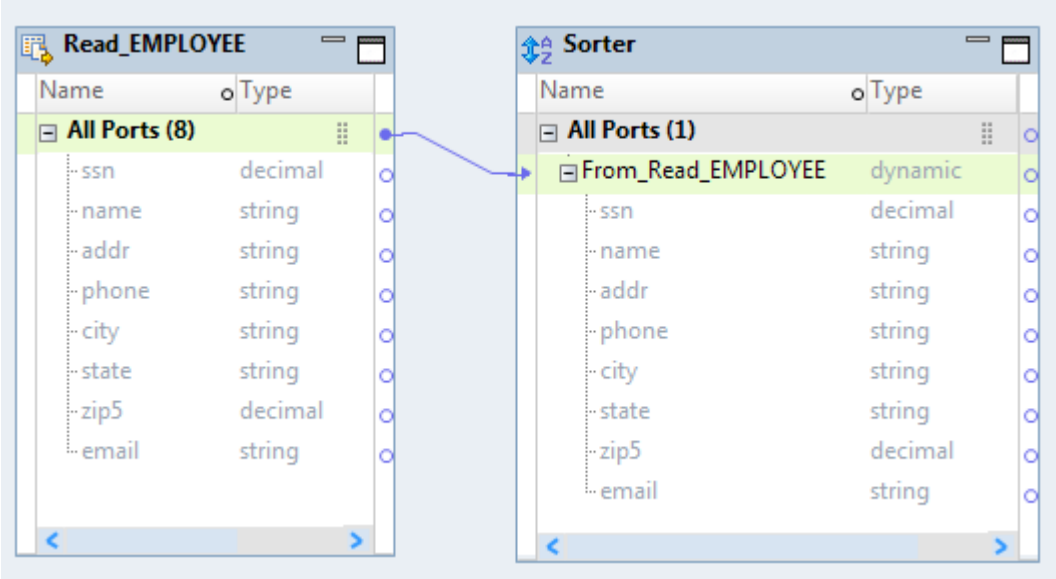
Display the generated ports based on the order of the ports in the Read transformation.

Reordering ports based on this option helps you retain the original order of ports in the source. However, if one or more mid-stream transformations have dynamic ports and static ports, the Developer tool displays the ports in the order that they appear in the nearest upstream transformation with static ports. This option is valid only if the mapping has a single pipeline.

Example - Reorder Generated Ports

An employee flat file source has many columns that change frequently. 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:

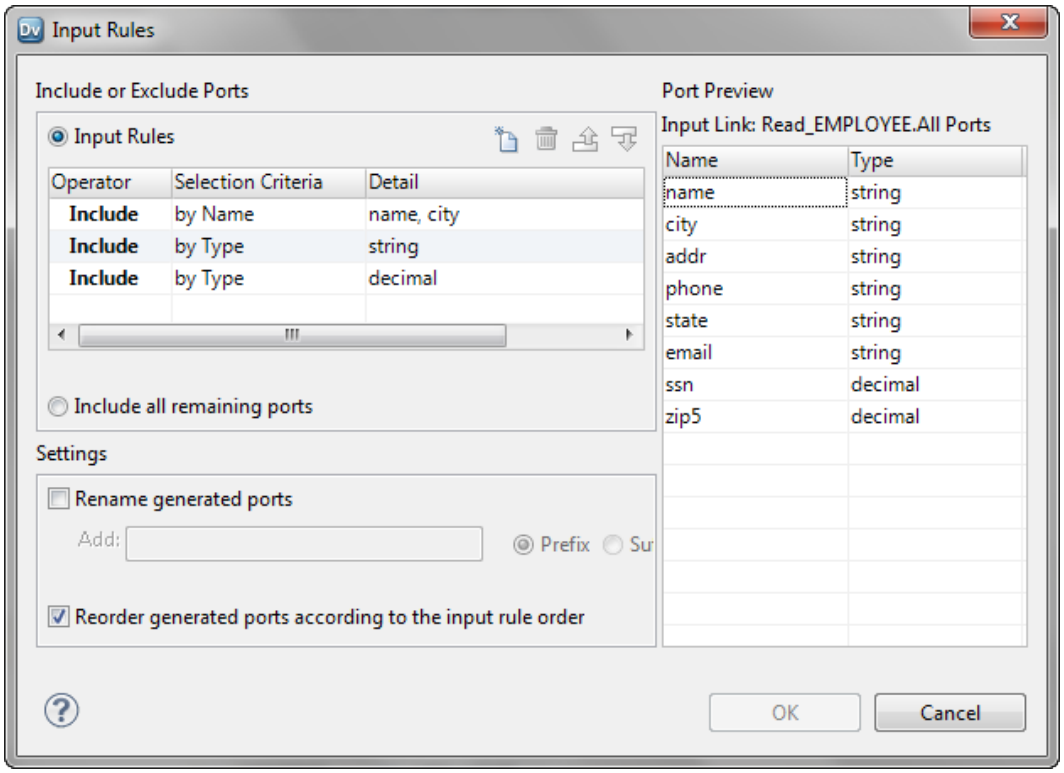


You configure the following input rules:

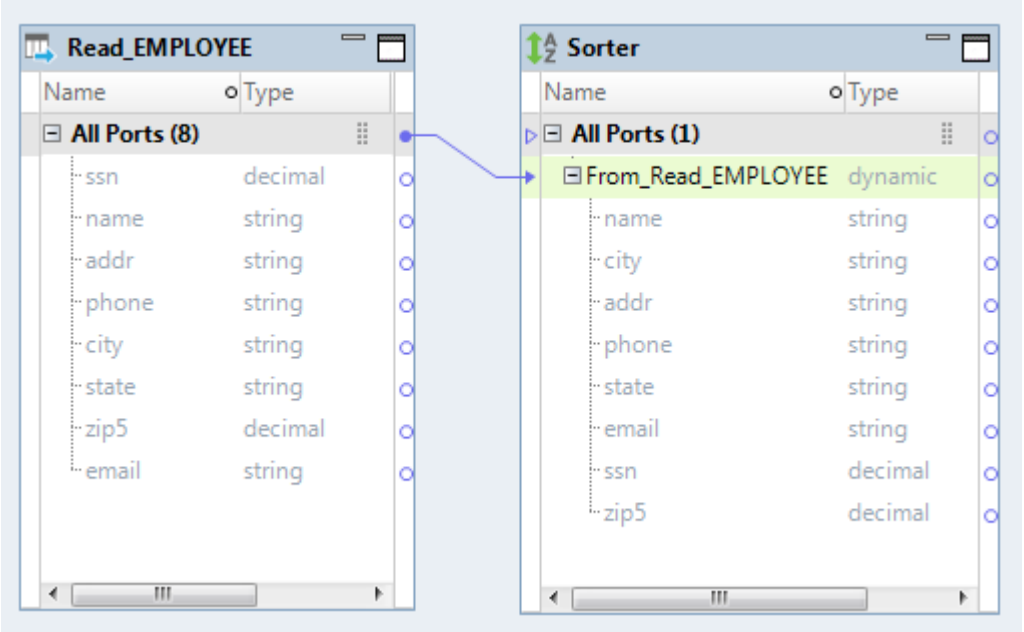
- Include ports by "name" and "city."
- Include all ports by string type.
- Include all ports by decimal type.

Then, you choose to reorder the ports based on the input rule order. To verify the order, you can preview the port order.

The following image shows the order of the input rules:



The following image shows the reordered generated ports based on the input rules settings.



Selection Rules and 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 a set of selection rules that determine ports. You reference a port selector in an expression. When the generated ports change in a dynamic mapping, the port selector can contain different ports in it. 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.

You can configure a port selector in the following mapping objects:

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 configure a port selector called Master-SelectorX that contains the ports A, B, and C. You configure Detail-SelectorY that contains the 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 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.

Write transformation

You can configure a port selector for the ports in the Write transformation.

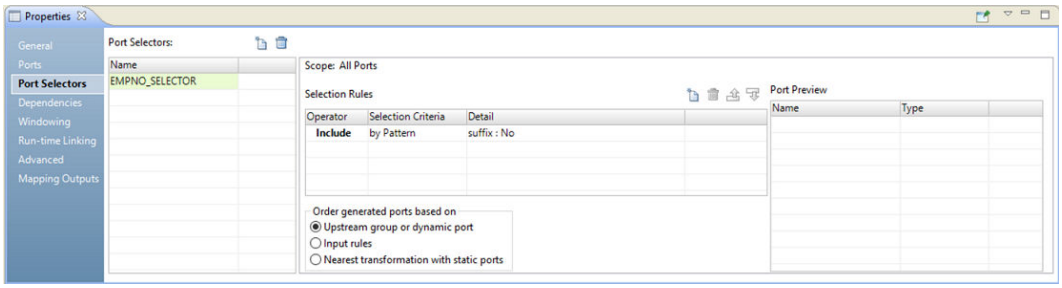
When you write data to a relational or Hive target, you can choose to create or replace the target table at run time. You can define a DDL query based on which the Data Integration Service must create or replace the target table at run time. You can also configure a port selector in the DDL query.

Port Selector Configuration

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 static 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.

Selection Rules

The selection rules associated with a port selector determine 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.

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 column name, port type, pattern, or complex data type definition. To include ports based on the column name, search for specific names or search for a pattern of characters in the name.

Detail

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:

Selection Criteria	Description	Detail
All	Includes all ports.	No details required.
Name	Filters ports based on the port name.	Select the port names from a list of values or use a parameter of type Port or Port List.
Type	Filters ports based on the data type of each port.	Select data types from a list.
Pattern	Filters ports by a string of characters in the name or by a regular expression.	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.
Complex Data Type Definition	Filters ports by a complex data type definition.	Choose prefix, suffix, or regular expression as the pattern type for the complex data type definition. Then, enter a value for the pattern or use a parameter of type String.

Example - Selection Rules and Port Selectors

You configure the mapping to use dynamic sources, but the column that contains salary information in each source file has a different name. The column names for the different sources are `Salary`, `Monthly_Salary`, or `Base_Salary`.

You perform the following tasks to run the expression with any of the salary port names:

1. You create a port selector named "Salary_PortSelector."
2. You create a selection rule to accept any port name with the suffix of "Salary."
3. Configure the expression to include the port selector name instead of a specific column name. The expression has the following syntax:

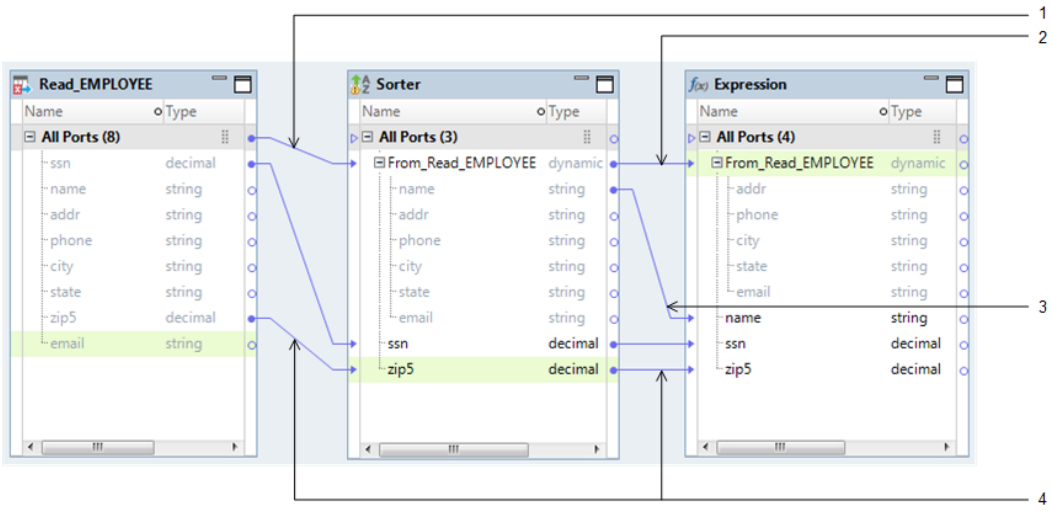
```
Salary_PortSelector * 12
```

Design-time Links

You can create different types of links when you design a dynamic mapping. You can create links between two ports, between a group and a dynamic port, between two dynamic ports, and from a generated port to a static port.

A design-time link (link) is a direct link that you create within a mapping. Transformations might change in such a way that you cannot create direct links when you design the mapping. If you cannot create links at design time, you can configure run-time links that determine the ports to link at run time.

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 static port
4. Link between two ports

You can create the following types of links when you design a mapping:

Link 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 previous 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 includes string ports.

Link 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 previous 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 a generated port to a static port.

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

For example, the previous 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.

Link two static ports.

Link ports between transformations in the same manner that you do for other mappings.

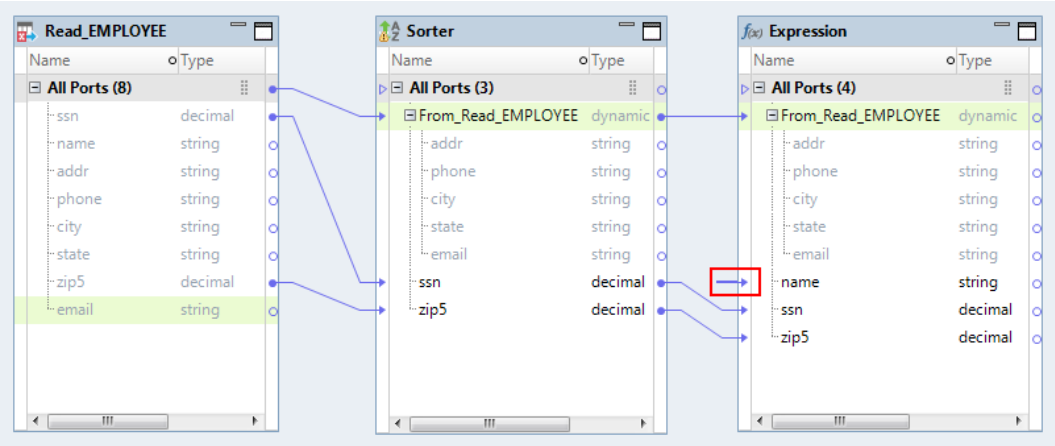
Link Resolution

Generated ports within a dynamic port can change based on the dynamic source or 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.

Run-time Links

A run-time link is a link between groups in which the ports might change at run time. The Data Integration Service determines the ports to link at run time based on policies and parameters.

Create a run-time link between groups of the mapping objects if the ports in the upstream transformation can change at run time. If the ports can change at run time, you cannot link the ports when you design the mapping. Create a run-time link that can use a parameter and a link policy to determine which ports to link at run time.

Create a run-time link in the following situations:

You configure the Read transformation to get columns from the data source or to use a source defined by a parameter.

For example, a Read transformation uses a parameter to change the sources or that 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.

You configure the Write transformation to get columns from the data source or the data object or to use a target defined by a parameter.

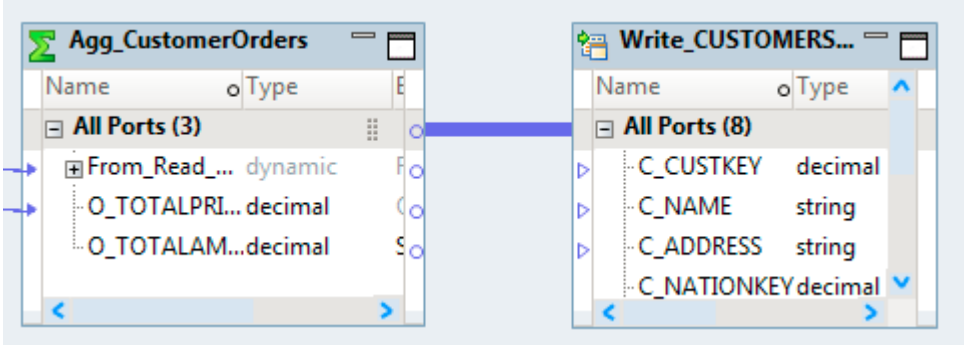
For example, a Write transformation defines columns based on an associated data object. The Write transformation uses a parameter to change the target or gets metadata changes from the target at run time. create and configure a run-time link to the Write transformation.

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

At run time, the Data Integration Service connects the ports based on the link policy or the parameter values and passes the data to the downstream port.

Create a run-time link between groups of transformations 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. Run-time links appear as thick lines in the mapping editor.

The following image shows a run-time link between an Aggregator transformation and a Write transformation:

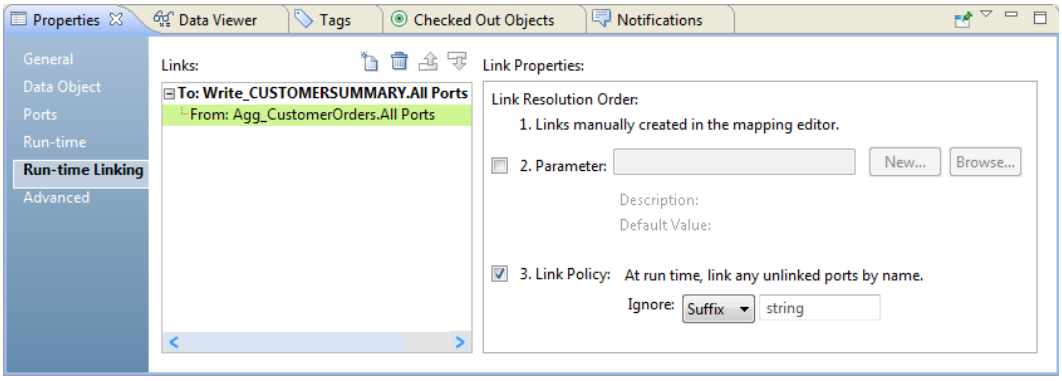


Run-time Link Configuration

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 to configure run-time link properties.

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



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

1. Links that you manually create in the mapping editor.

2. Links based on the parameter that you configured for a run-time link.
3. Links based on the link policy that you configured for a run-time link.

Configure the following properties for run-time links:

Choose the transformations

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 a parameter

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.

For example, you create a parameter named `Cust_InputLinkSet` of type new Input Link Set and provide the default value as follows: `C_Name -> Cust_name`. At run time, Data Integration Service creates a link between the ports `C_Name` and `Cust_name`. You can change the values of the parameter for the next mapping run as follows: `CustFirstName -> Cust_name`

Configure the link policy

A link policy links any unlinked ports by name. When you define target columns by mapping flow, the mapping propagates all ports from the source or from upstream objects. Use a run-time linking policy to propagate ports of certain types or with certain names. 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.

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.

The linking properties appear in the **Run-time Linking** dialog box or on the **Run-time Linking** tab in the **Properties** based on the action you take to link the groups:

- 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.
- If you cannot create dynamic ports, you can drag a group from an upstream transformation. Then, select **Create Run-time Link** in the **Linking** dialog box to open the **Run-time Linking** dialog box.
- Select the **Run-time Linking** tab in the **Properties** view of the downstream transformation.

Example - Run-time Links

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:

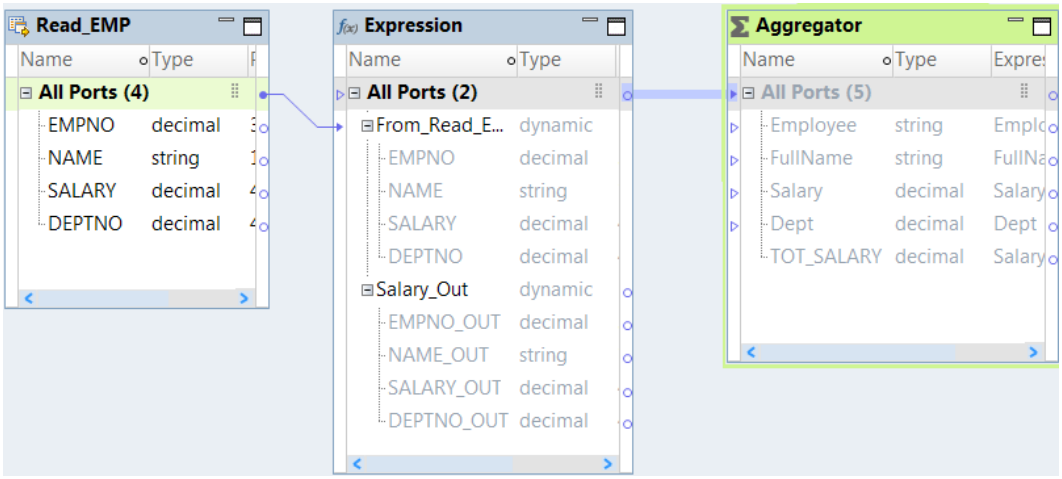
```
EMPNUM_OUT
FULLNAME_OUT
```

```
SALARY_OUT
DEPT_OUT
```

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:



Troubleshooting Dynamic Mappings

Consider the following troubleshooting tips when you design and test dynamic mappings:

The dynamic ports in my mapping include a column with XML data type, and the mapping failed.

You cannot propagate XML data through a mapping if any of the following conditions are true:

- You configure the Read or Write transformation to get columns from the data source.
- You configure the Write transformation to get columns from the data flow.
- You configure the data obect for the target to create or replace the target at run time.

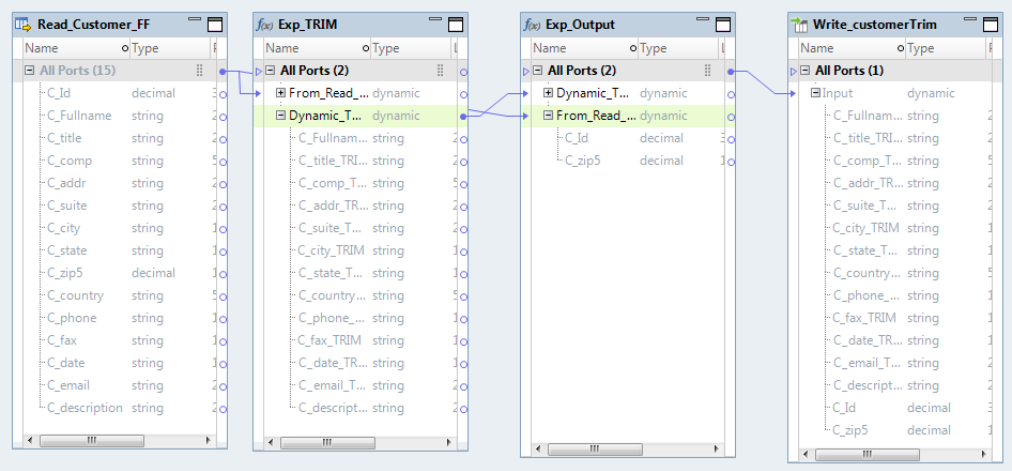
My dynamic mapping failed with run-time errors for parameters and links. I would like to make sure that the parameters and links successfully resolve before I run the mapping.

When you run a dynamic mapping, the Data Integration Service compiles the mapping to complete the following tasks:

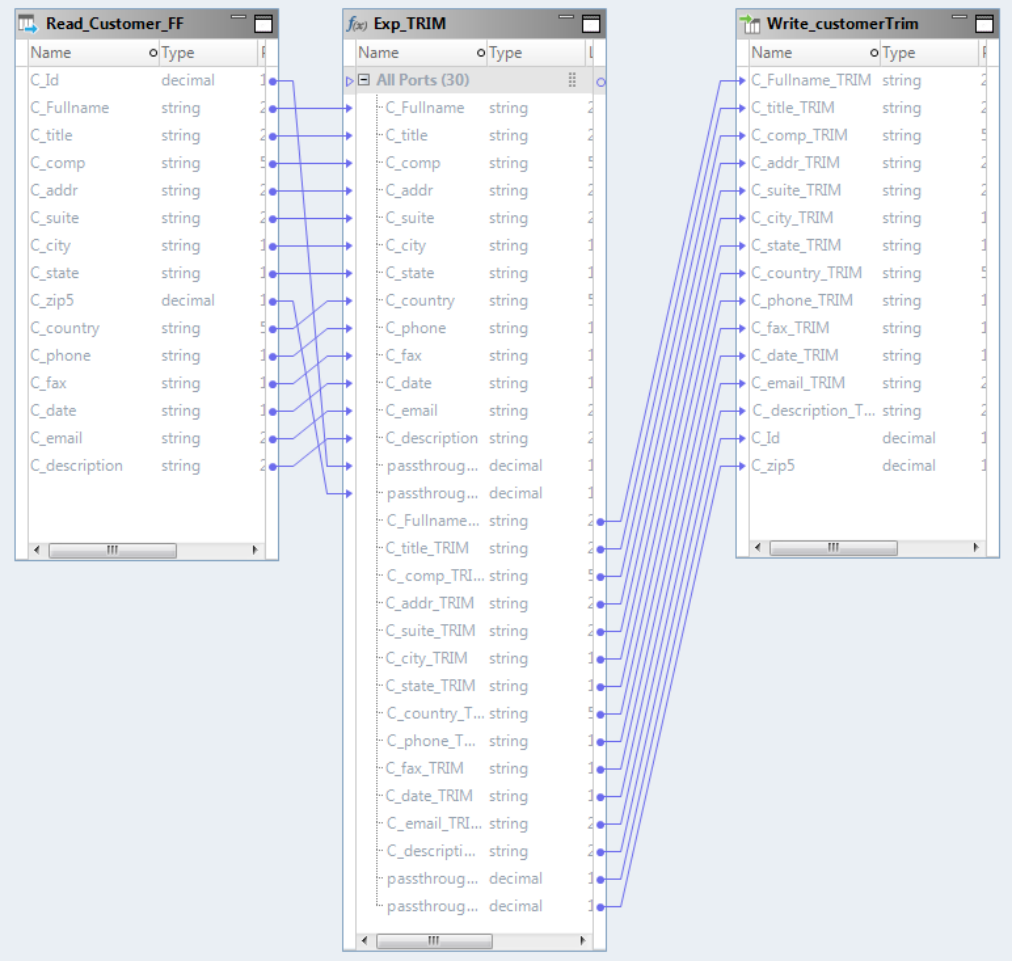
- Resolve parameter values.
- Expand dynamic ports and convert generated ports to static ports.
- Link static ports.
- Resolve run-time links to connect the ports.

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 following image shows the compiled version of the dynamic mapping in which the generated ports are converted to static ports and are linked:



CHAPTER 8

How to Develop and Run a Dynamic Mapping

This chapter includes the following topics:

- Developing and Running Dynamic Mappings, 160
- Configuring a Dynamic Source, 161
- Creating a Dynamic Port, 163
- Configuring Dynamic Ports Using Input Rules, 164
- Creating a Port Selector, 168
- Creating a Dynamic Expression , 169
- Configuring a Dynamic Target, 171
- Creating and Configuring a Run-time Link, 176
- Validating a Dynamic Mapping, 178
- Validating Dynamic Sources and Targets, 178
- Running a Dynamic Mapping, 179

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.

Task	Reference
Create a mapping and add mapping objects.	"Creating a Mapping" on page 38 "Adding Objects to a Mapping" on page 38
Configure dynamic sources for the Read or Lookup transformations to get metadata changes from flat file or relational sources at run time.	"Configuring a Dynamic Source" on page 161

Task	Reference
Create dynamic ports in transformations and link ports.	"Creating a Dynamic Port" on page 163
Define input rules for dynamic ports to determine which generated ports to create. - Define input rules to include or exclude ports. - Rename the generated ports. - Optionally, reorder generated ports.	"Configuring Dynamic Ports Using Input Rules" on page 164
Configure the transformations.	Refer to the *Informatica Developer Transformation Guide* for details to configure transformations in the mapping.
Optionally, create port selectors to use in the transformation logic of Joiner, Lookup, or Expression transformations.	"Creating a Port Selector" on page 168
Optionally, create dynamic expressions to use in Expression transformations.	"Creating a Dynamic Expression " on page 169
Configure Write transformations to write to dynamic targets as follows: - 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. - Select a target schema strategy for the Write transformations that represent relational and Hive targets.	"Configuring a Dynamic Target" on page 171
Create and configure a run-time link to determine which ports to link at run time.	"Creating and Configuring a Run-time Link" on page 176
After you determine where you want to use parameters in the mapping, create and assign parameters. - Configure sources as parameters - Configure targets as parameters - Configure transformation properties as parameters	"How to Configure Parameters" on page 68
Validate the mapping.	"Validating a Dynamic Mapping" on page 178
Validate synchronized data sources and targets in the dynamic mapping.	"Validating Dynamic Sources and Targets" on page 178
Compile and run the dynamic mapping.	"Running a Dynamic Mapping" on page 179

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.

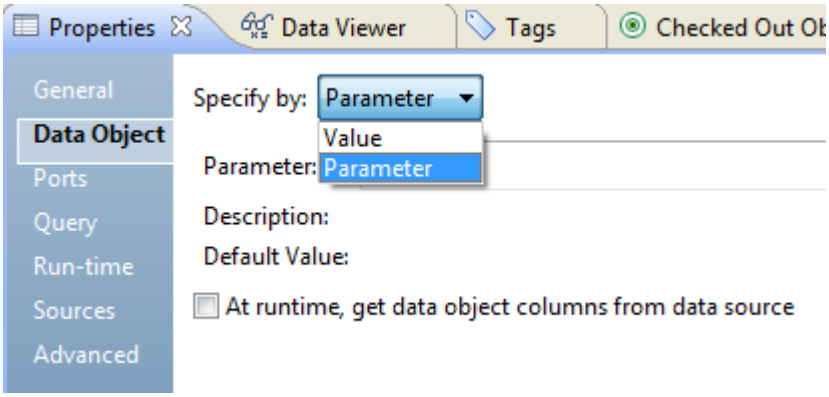
Configure Read and Lookup transformations to get metadata at run time

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.



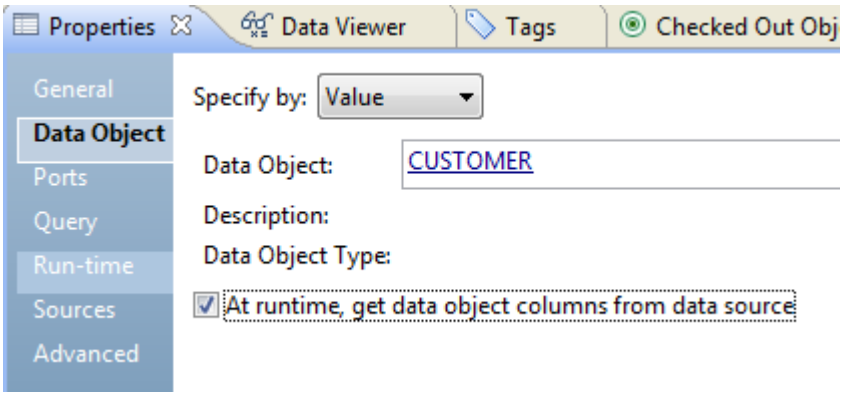
4. Click **New** to create a new parameter, or **Browse** to choose an existing parameter.

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**.



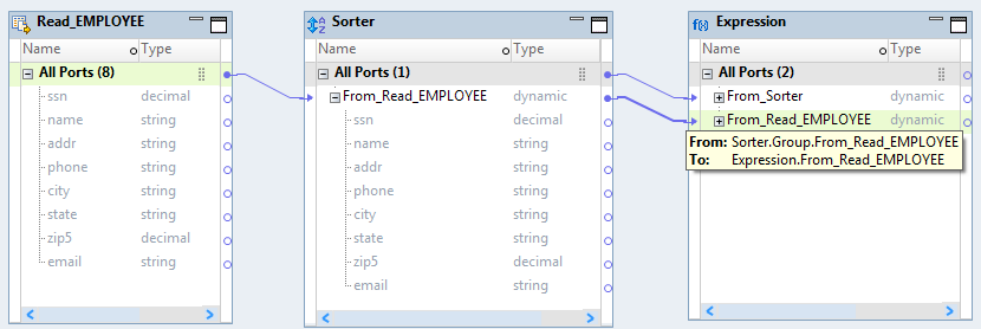
Creating a Dynamic Port

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.

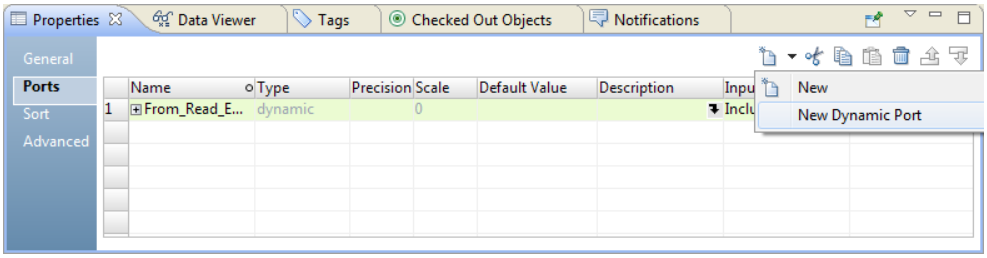
1. Create a dynamic port in the following ways:
   - Drag the All Ports group or a dynamic port from another transformation.

     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:

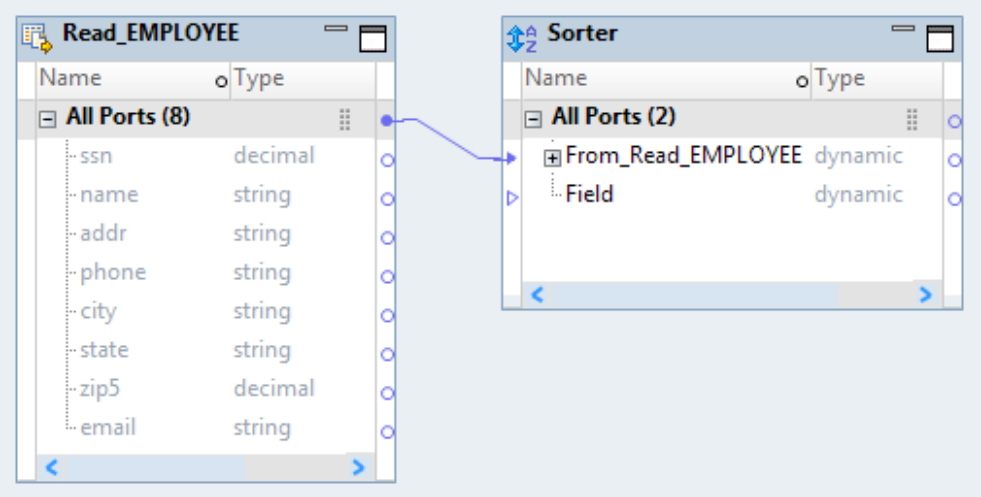


   - In the **Properties** view of the transformation, select **New Dynamic Port** on the **Ports** tab.



     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:



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 paramter.

   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**.

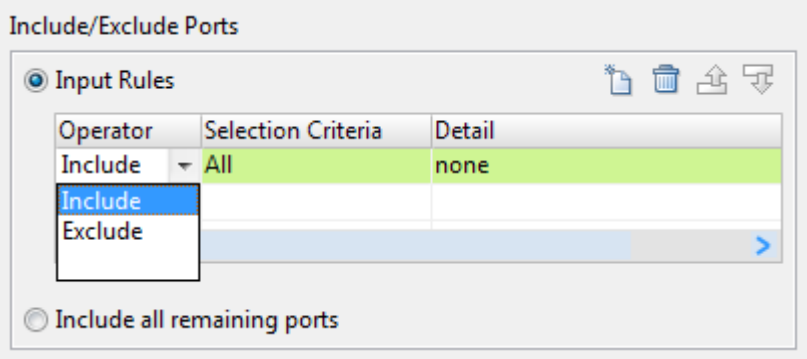
   This rule includes ports from the upstream transformation that are not part of other dynamic ports.

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.

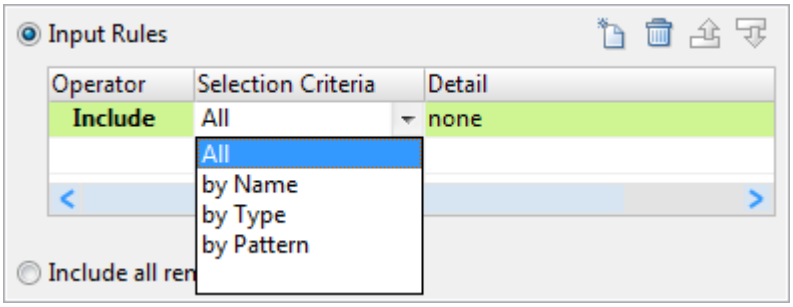
1. From the **Operator** column, choose the **Include** or **Exclude** operator.

   The operator determines whether the rule must include or exclude ports.



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

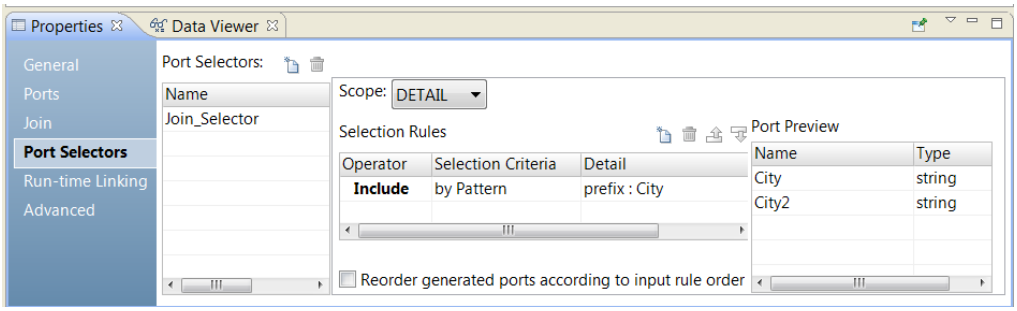
Create a port selector to determine which ports to use in a dynamic expression, a lookup condition, or a joiner condition.

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:



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**.

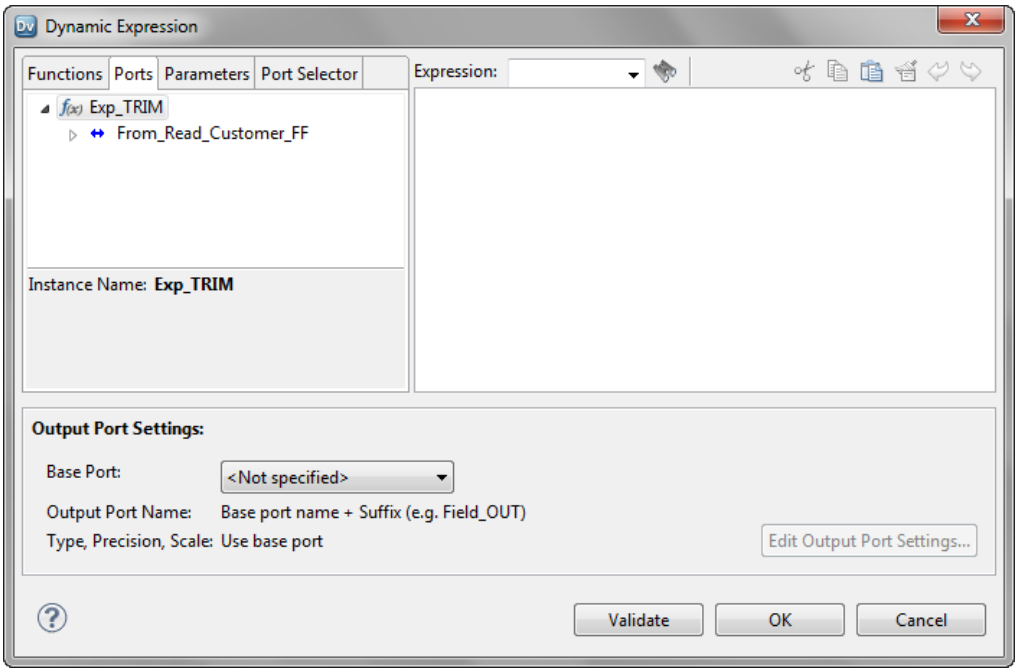
   The Developer tool creates a dynamic port with default properties.

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.

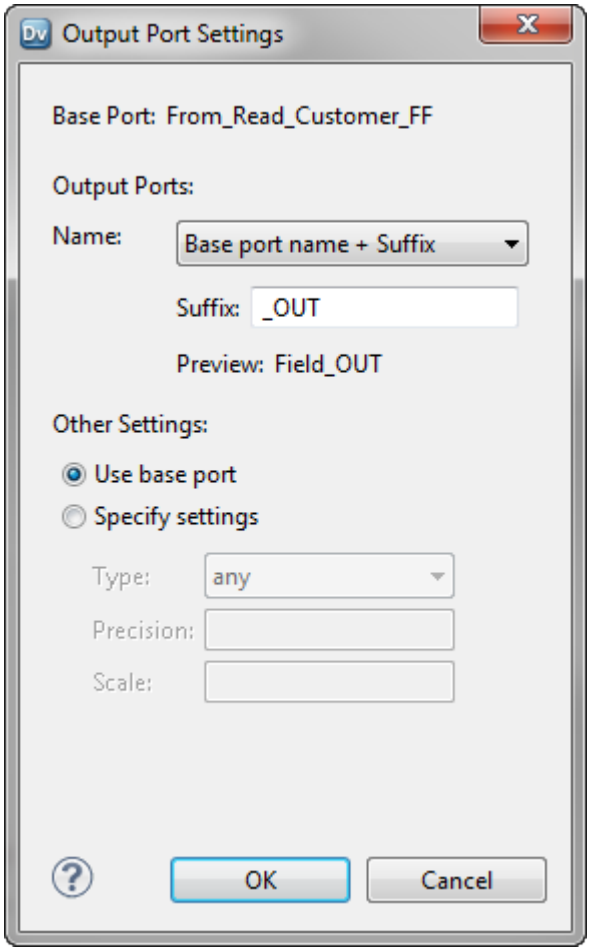
   For example, `LTRIM(RTRIM(Dynamic_Customer))`, where Dynamic_Customer is a dynamic port.

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:
   a. Click **Edit Output Port Settings**.

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



   b. 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.
   c. 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.
   d. 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.

Select a target schema strategy

Select a target schema strategy to either retain the existing target schema or create or replace the target table at run time.

When you configure the Write transformation to create or replace the target, by default, the Data Integration Service drops any existing target table associated with the write object. The Data Integration Service then creates a table based on the mapping flow or data object.

If you want to customize the table or specify additional parameters such as partitions, you can define a DDL query based on which the Data Integration Service must create or replace the target table. The table contains the columns that you define in the DDL query.

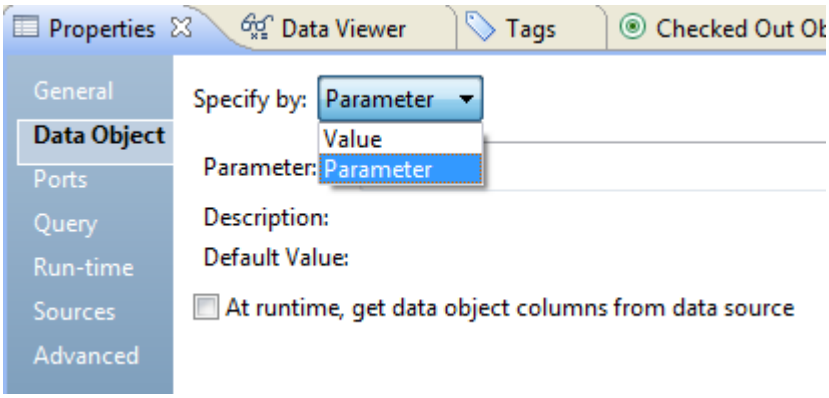
Define Write transformation ports from mapping flow

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.

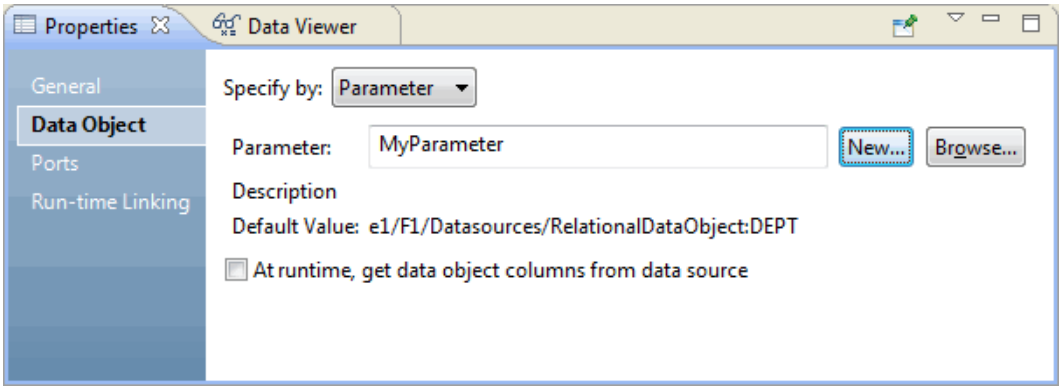
1. Select the Write transformation in the mapping editor.
2. In the **Properties** view, click the **Data Object** tab.
3. Select **Parameter** in the **Specify by** list.



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:

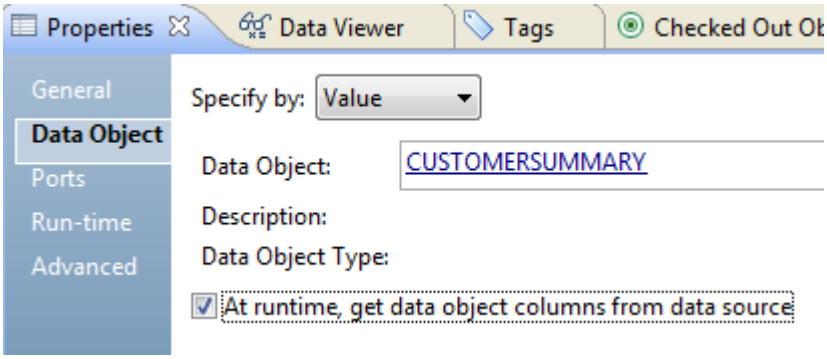


Getting Target Object Columns from the Data Source at Run-Time

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 a DDL Query to Create or Replace the Target at Run Time

When you select the Target Schema Strategy option to create or replace the target at run time, you can define a DDL query based on which the Data Integration Service must create or replace the target table at run time. You can define a DDL query for relational and Hive targets. You can enter placeholders and parameters in the DDL query.

1. In the **Properties** view, click the **Advanced** tab.
2. Select **Create or replace table at runtime** from the **Target Schema Strategy** list.

   The **DDL query** field is available.
3. Click **Edit**.

   The **DDL query** dialog box appears.
4. Enter the DDL query in the editor.

You can enter placeholders in the DDL query. The Data Integration Service substitutes the placeholders with the actual values at run time. For example, if a table contains 50 columns, instead of entering all the column names in the DDL query, you can enter a placeholder.

You can enter the following placeholders in the DDL query:

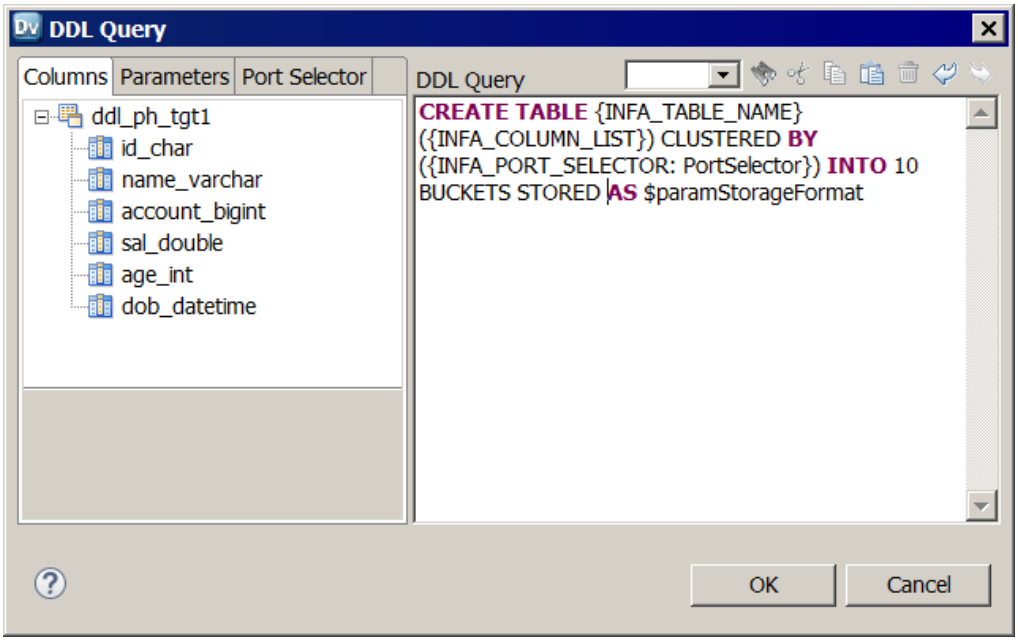
- INFA_TABLE_NAME. Fetches the target table name at run time.
- INFA_COLUMN_LIST. Fetches a list of columns in the target table at run time.
- INFA_PORT_SELECTOR. Adds port selectors.

Note: The placeholder names are case sensitive. You must enclose the placeholders within two curly brackets. For example, {INFA_TABLE_NAME}.

You can also perform the following steps to define the DDL query.

- To add a column name, double-click a column in the **Columns** tab.
- To define a parameter, click the **Parameters** tab, and double-click a parameter name. You can also click **Manage Parameters** to add, edit, or delete parameters.
- To configure a port selector, click the **Port Selector** tab, and double-click a port selector. You can also click **New** to configure a new port selector.

The following image shows a DDL query to create a Hive target table:



The DDL query in the image contains the INFA_TABLE_NAME, INFA_COLUMN_LIST, and INFA_PORT_SELECTOR placeholders. It also contains a parameter to define the storage format.

If you do not enter a DDL query, the Data Integration Service creates the target based on the mapping flow or data object.

5. Click **OK** to save the DDL query.

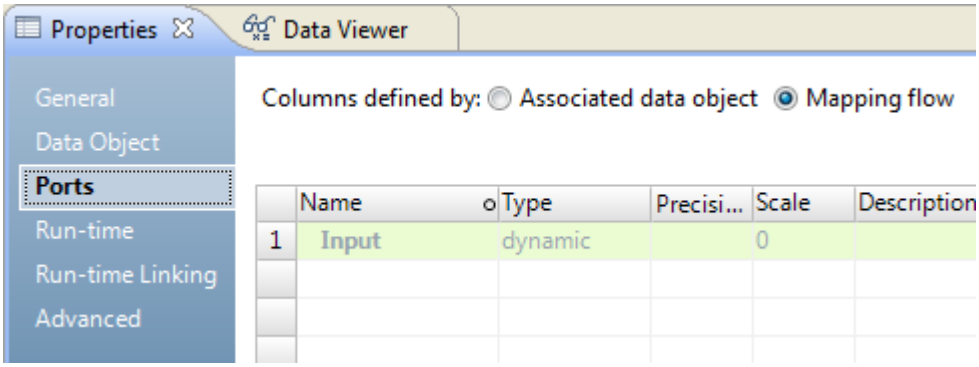
Defining Write Transformation Ports

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**.

   The following image shows the **Ports** tab populated with ports defined by the associated data object:



3. Enable dynamic ports and targets:

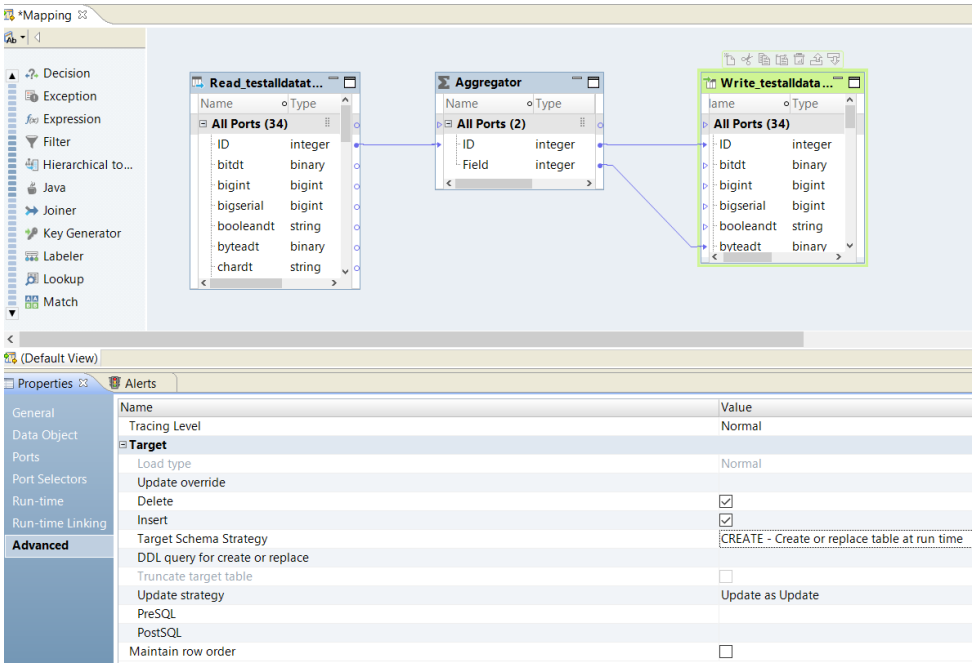
   a. Drag upstream ports to the **input** pane of the Write transformation.

      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** in the **Target Schema Strategy** list.

      The following image shows the **Target Schema Strategy** option in the **Advanced** tab of the target object:



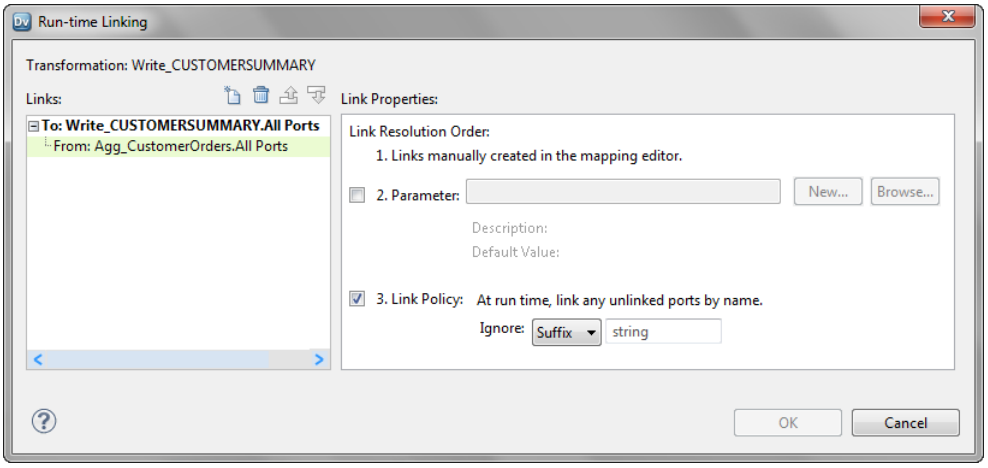
      At run-time, the Data Integration Service retains, or drops and replaces, the target table.

      Note: When a mapping contains multiple targets whose columns are defined by the same physical data object, select the **CREATE - Create or replace table at run time** target schema strategy option for only one of the targets. If you select the 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.

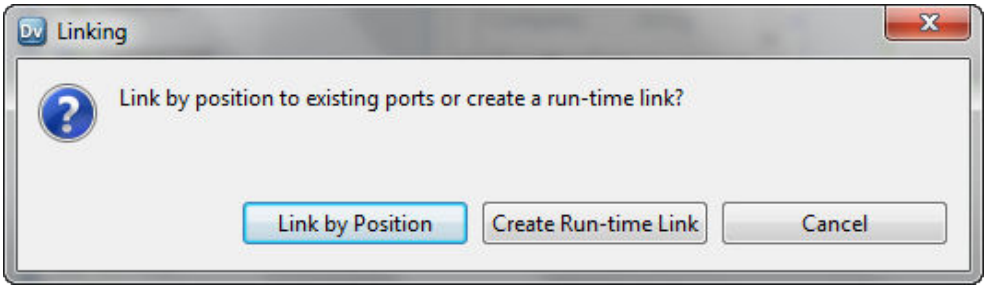
Creating and Configuring a Run-time Link

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.

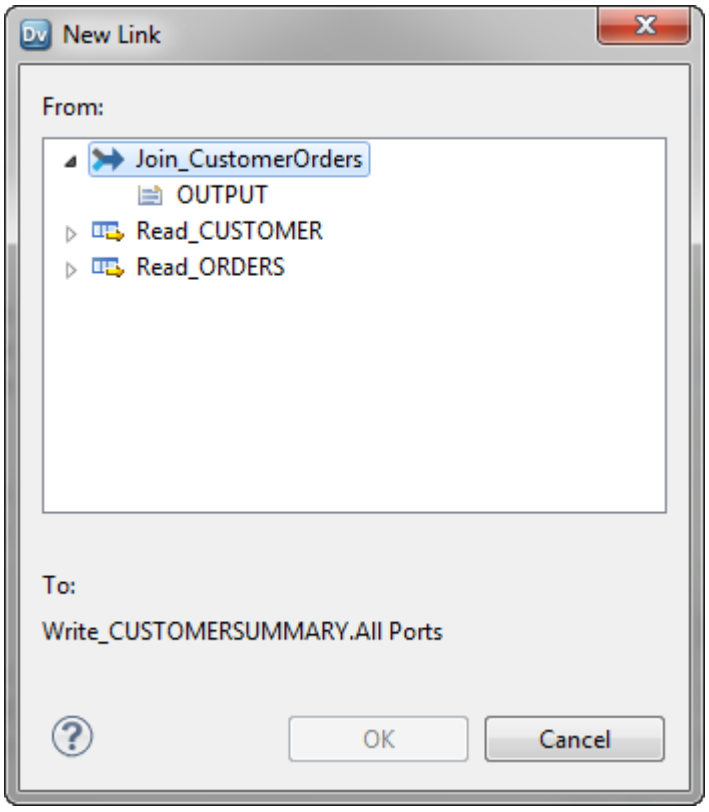
c. Optionally, create new parameters or edit parameters from this dialog box.

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.

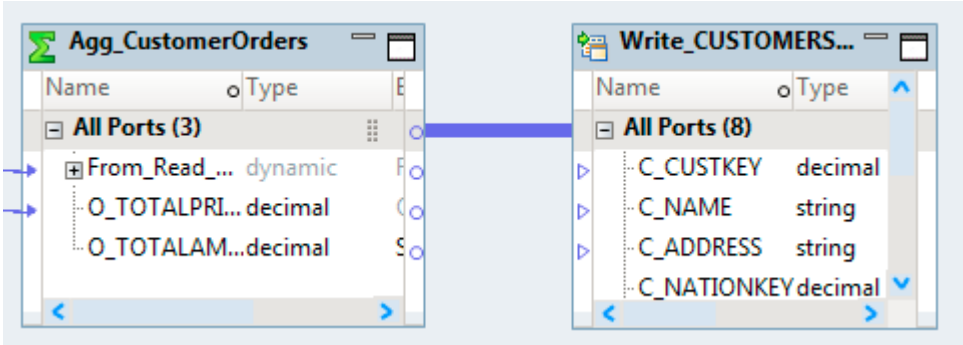
The **New Link** dialog box appears.



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 a Dynamic Mapping

Validate a mapping to ensure that the Data Integration Service can read and process the entire mapping.

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. Re-validate the mapping after the source schema has changed or after you changed the parameter values.

Validating Dynamic Sources and Targets

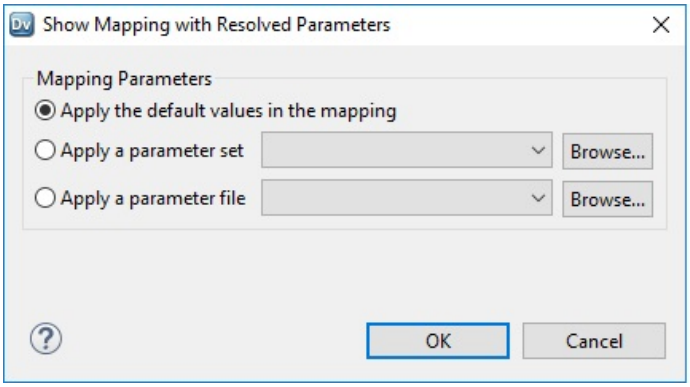
Validate the sources and targets in a dynamic mapping to ensure that the Data Integration Service can read and process the entire mapping when a source or target synchronizes with a relational database at run time.

To validate the sources and targets, resolve the mapping parameters and validate the generated instance of the mapping that contains the resolved parameters. The mapping must be valid before you resolve the mapping parameters.

Note: To resolve mapping parameters, parameters must be configured in the mapping. If parameters are not configured, you cannot validate the dynamic sources and targets.

1. In the Developer tool, right-click a mapping in the editor or the Object Explorer view. Select **Show Mapping with Resolved Parameters**.

   The **Show Mapping with Resolved Parameters** dialog box appears.



2. Select one of the following mapping parameters options:
   - Apply the default values in the mapping. The Data Integration Service applies the default parameter values configured in the mapping.
   - Apply a parameter set. The Data Integration Service applies the parameter values configured in the parameter set.
   - Apply a parameter file. The Data Integration Service applies the parameter values configured in the parameter file.
3. Click **OK**.

   The Developer tool generates the run-time instance of the mapping. The run-time mapping appears in a new tab.
4. Right-click the run-time instance of the mapping and select **Validate**.

   If errors appear in the **Validation Log** view, fix the errors and validate the mapping again.
5. When the mapping is valid, click **File** > **Save** to save the mapping.
6. Re-validate the mapping after the source schema has changed or after you change the parameter values.

Running a Dynamic Mapping

Run the dynamic mapping to write the transformed data to the target.

1. 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.
2. Click **Window** > **Show View** > **Progress** to view the progress of the mapping run.

   The **Progress** view opens.
3. Change parameter values between mapping runs.
4. Re-run the mapping after the source schema has changed or after you change the parameter values.

CHAPTER 9

Dynamic Mapping Use Cases

This chapter includes the following topics:

- Use Case: Dynamic Mapping for Metadata Changes in Relational Sources, 180
- Use Case: Reuse Dynamic Mapping for Different Sources and Targets, 191

Use Case: Dynamic Mapping for Metadata Changes in Relational Sources

You are a developer for an organization that must aggregate total customer orders. The 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. You need to develop a dynamic mapping that can accommodate the changing source schema and aggregate the total customer orders.

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:

Name	Native Type	Precision	Scale
C_CUSTKEY	number(p,s)	38	0
C_NAME	varchar2	25	0
C_ADDRESS	varchar2	40	0
C_NATIONKEY	number(p,s)	38	0
C_PHONE	varchar2	15	0
C_ACCTBAL	number(p,s)	10	2
C_MKTSEGMENT	varchar2	10	0

The following table lists the columns and metadata for the ORDERS table:

Name	Native Type	Precision	Scale
O_ORDERKEY	number(p,s)	38	0
O_CUSTKEY	number(p,s)	38	0
O_ORDERSTATUS	varchar2	1	0
O_TOTALPRICE	number(p,s)	10	2
O_ORDERDATE	date	19	0
O_ORDERPRIORITY	varchar2	15	0
O_CLERK	varchar2	15	0
O_SHIPPRIOIRITY	number(p,s)	30	0

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:

Name	Native Type	Precision	Scale
C_CUSTKEY	number(p,s)	38	0
C_NAME	varchar2	25	0
C_ADDRESS	varchar2	40	0
C_NATIONKEY	number(p,s)	38	0
C_PHONE	varchar2	15	0
C_ACCTBAL	number(p,s)	10	2
C_MKTSEGMENT	varchar2	10	0
C_TOTALAMOUNT	number(p,s)	10	2

Dynamic Mapping

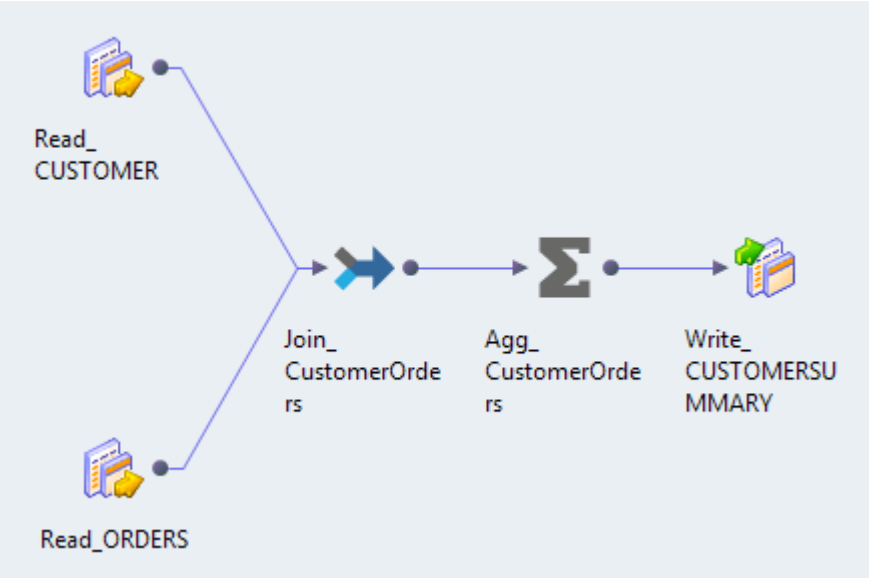
Create a mapping m_CustomerLoad and configure the following dynamic mapping functionality:

- 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:

1. Fetches the structure of the data objects and metadata changes in the source files.
2. Passes the new and changed columns to each transformation through dynamic ports.
3. Connects the new and changed ports to the Write transformation.
4. Writes the transformed data to the target.

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.

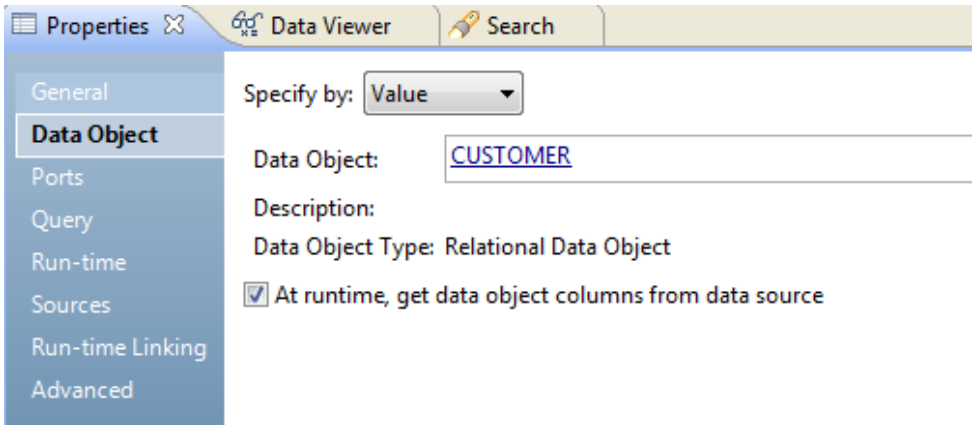
Step 1. Configure the Read Transformations

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:



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.

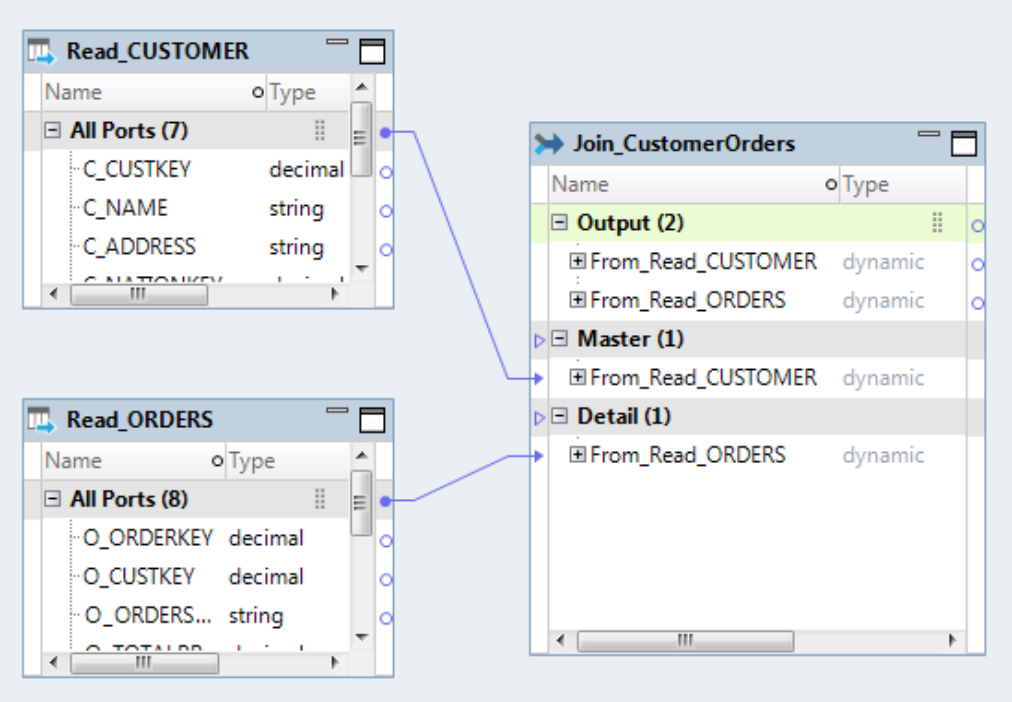
1. Add a Joiner transformation Join_CustomerOrders to the mapping.
2. Create dynamic ports in the Joiner transformation:
   a. From the Read_Customer transformation, drag the All Ports group to the Master group in the Joiner transformation.

      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.

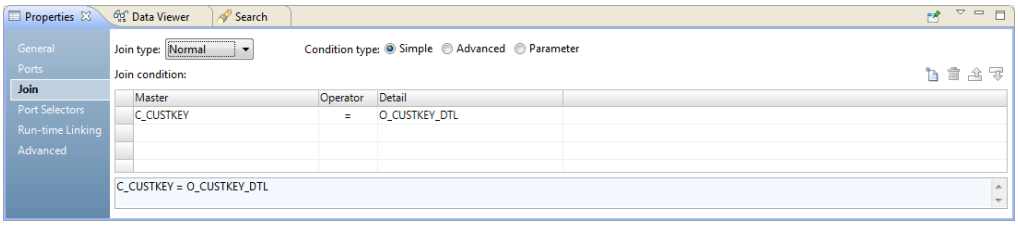
   The dynamic ports include all ports from the corresponding Read transformations as generated ports.

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

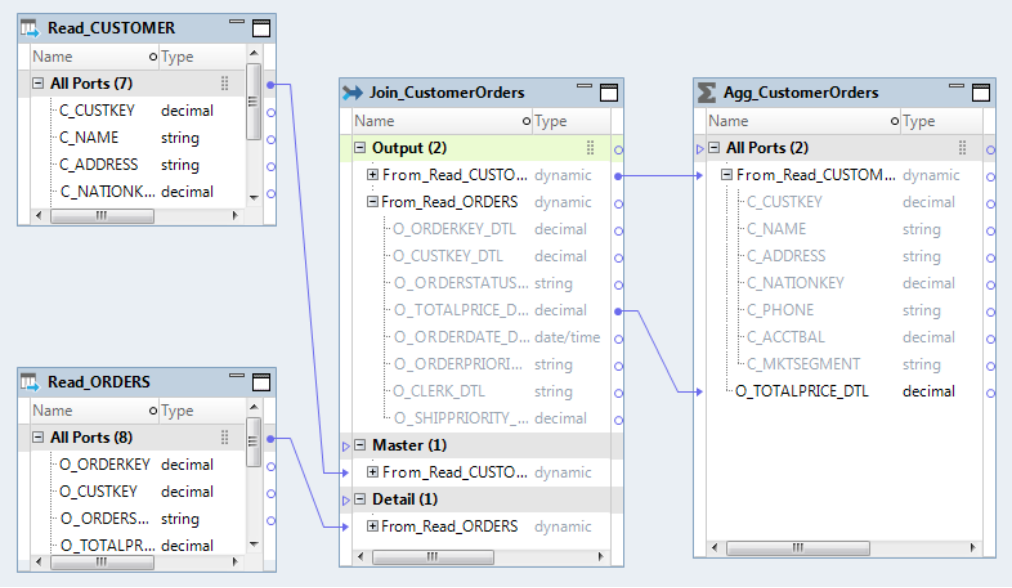
Add an Aggregator transformation to the mapping and configure dynamic ports to receive any new and changed columns from the Joiner transformation. Create an aggregate expression to calculate the total price of customer orders and group the aggregation by customer.

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.

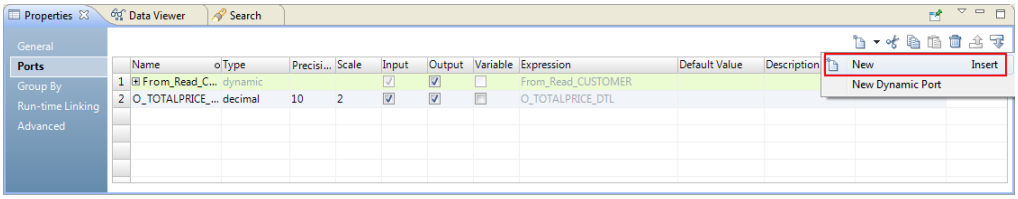
      A dynamic port From_Read_CUSTOMER appears in the Aggregator transformation.

   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:



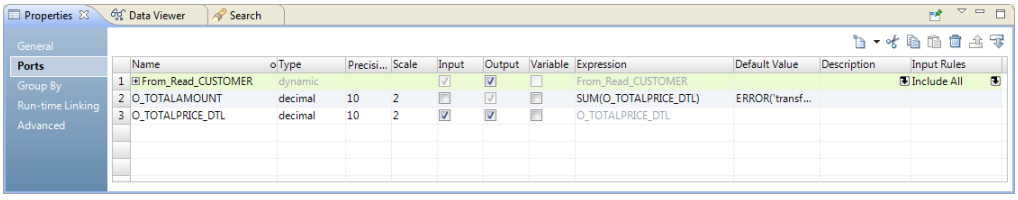
3. In the **Properties** view, click the **Ports** tab.
4. Click the **New** button to create a port to aggregate prices of the orders.



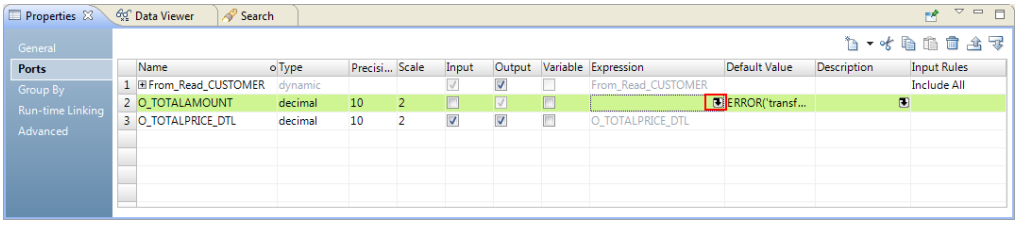
The Developer tool creates a new port called Field.

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:

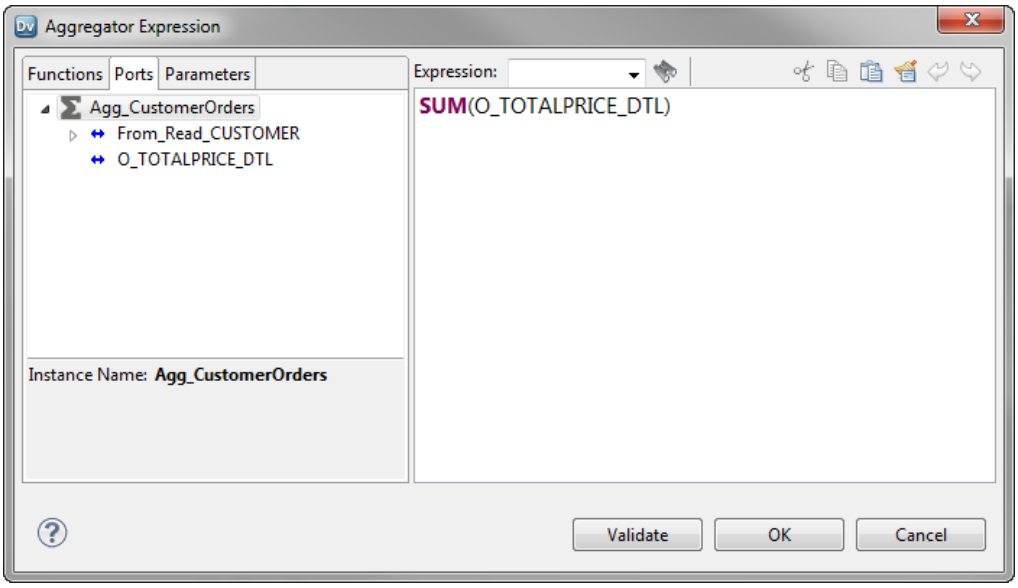


6. In the Expression column for the O_TOTALAMOUNT port, click the **Open** button.



The **Aggregator Expression** window appears.

7. Replace the existing expression in the editor with the following expression: `SUM(O_TOTALPRICE_DTL)`

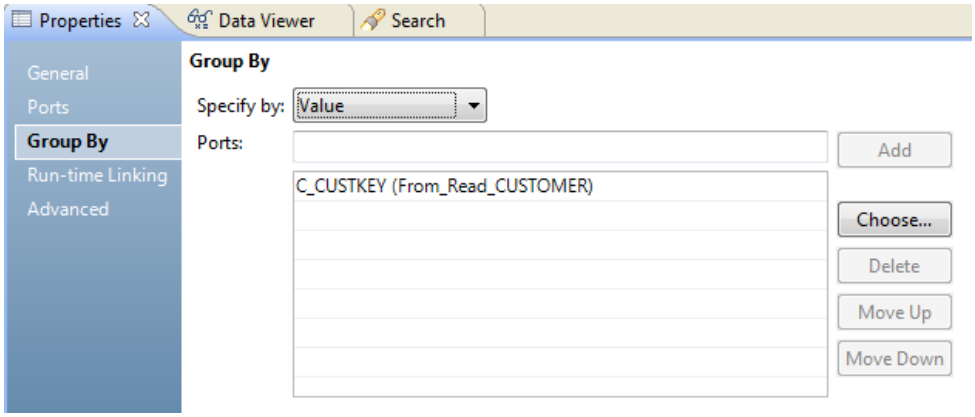


8. Click **Validate** to validate the expression.
9. Click **OK**.
10. Click **OK** to exit the **Aggregator Expression** editor.
11. In the **Properties** view, click the **Group By** tab.
12. Specify the group by port to aggregate the total price by market segment as follows:
    a. Make sure that **Value** from the **Specify by** list is selected.
    b. Click **Choose**.

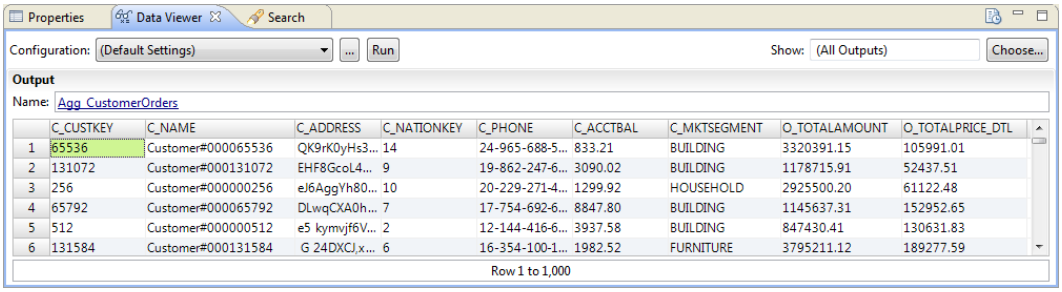
       The **Ports** dialog box appears.

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_CUSTOMERSUMMARY.

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**.

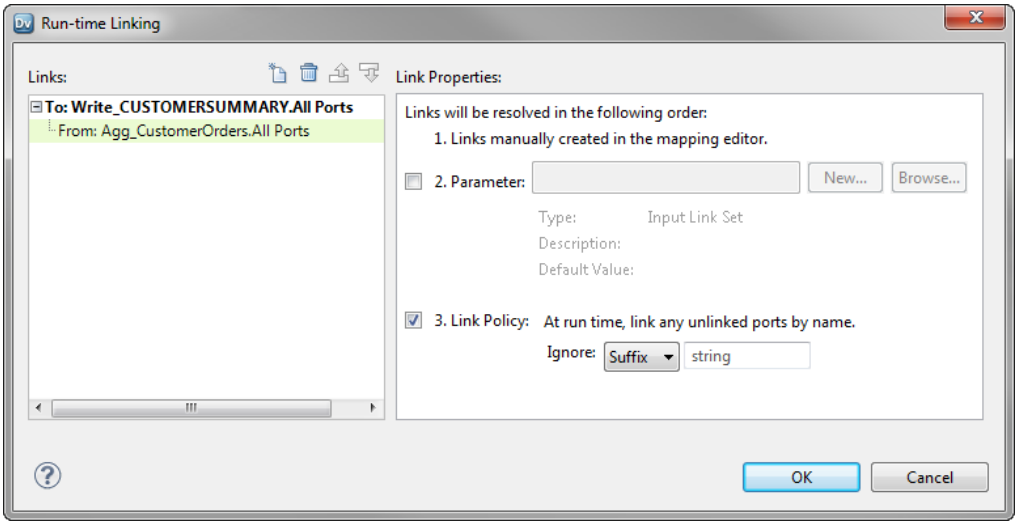
Step 5. Create and Configure a Run-time Link

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.

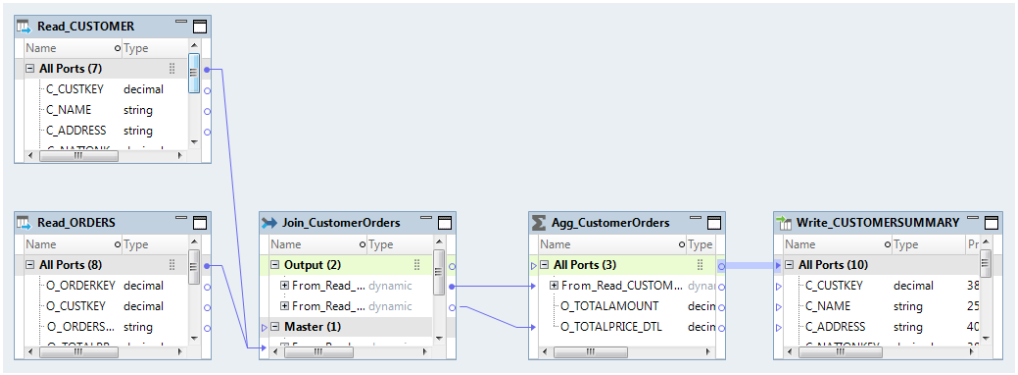
   The **Run-time Linking** dialog box appears.

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.

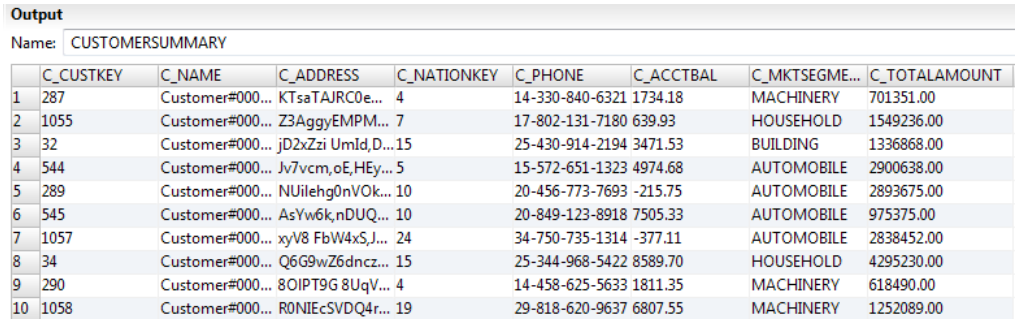
5. Click **Window** > **Show View** > **Data Viewer**.

   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.



Output

Name: CUSTOMERSUMMARY

	C_CUSTKEY	C_NAME	C_ADDRESS	C_NATIONKEY	C_PHONE	C_ACCTBAL	C_MKTSEGME...	C_TOTALAMOUNT
1	287	Customer#000...	KTsaTAJRC0e...	4	14-330-840-6321	1734.18	MACHINERY	701351.00
2	1055	Customer#000...	Z3AggyEMPM...	7	17-802-131-7180	639.93	HOUSEHOLD	1549236.00
3	32	Customer#000...	jD2xZzi UmId,D...	15	25-430-914-2194	3471.53	BUILDING	1336868.00
4	544	Customer#000...	Jv7vcm,oE,HEy...	5	15-572-651-1323	4974.68	AUTOMOBILE	2900638.00
5	289	Customer#000...	NUilehg0nVOk...	10	20-456-773-7693	-215.75	AUTOMOBILE	2893675.00
6	545	Customer#000...	AsYw6k,nDUQ...	10	20-849-123-8918	7505.33	AUTOMOBILE	975375.00
7	1057	Customer#000...	xyV8 FbW4xS,J...	24	34-750-735-1314	-377.11	AUTOMOBILE	2838452.00
8	34	Customer#000...	Q6G9wZ6dncz...	15	25-344-968-5422	8589.70	HOUSEHOLD	4295230.00
9	290	Customer#000...	8OIPT9G 8UqV...	4	14-458-625-5633	1811.35	MACHINERY	618490.00
10	1058	Customer#000...	R0NIEcSVDQ4r...	19	29-818-620-9637	6807.55	MACHINERY	1252089.00

Step 7. Run the 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:

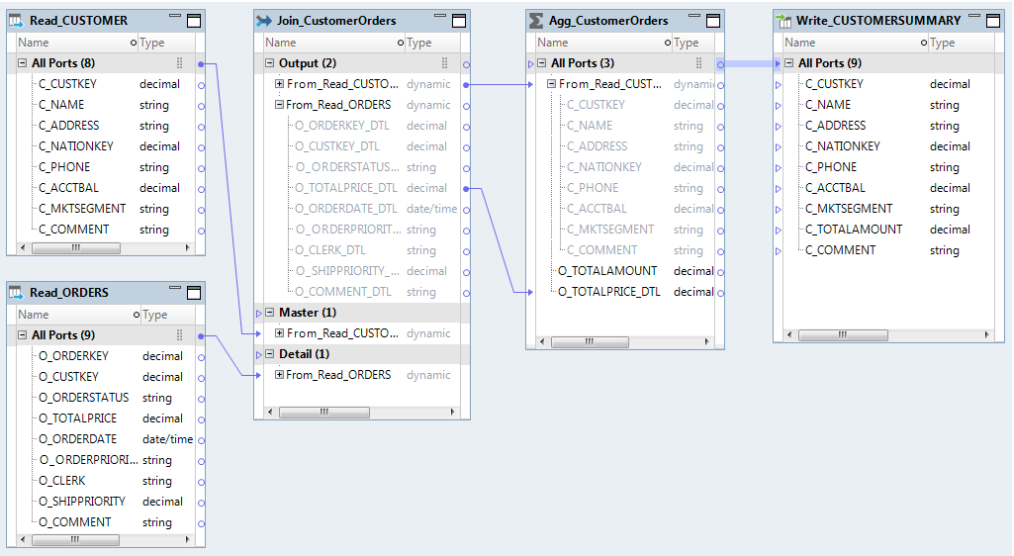
Name	Native Type	Precision	Scale
C_CUSTKEY	number(p,s)	38	0
C_NAME	varchar2	25	0
C_ADDRESS	varchar2	40	0
C_NATIONKEY	number(p,s)	38	0
C_PHONE	varchar2	15	0
C_ACCTBAL	number(p,s)	10	2
C_MKTSEGMENT	varchar2	10	0
C_COMMENT	varchar2	117	0

The following table lists the columns and metadata for the updated ORDERS table with the new O_COMMENT column:

Name	Native Type	Precision	Scale
O_ORDERKEY	number(p,s)	38	0
O_CUSTKEY	number(p,s)	38	0
O_ORDERSTATUS	varchar2	1	0
O_TOTALPRICE	number(p,s)	10	2
O_ORDERDATE	date	19	0
O_ORDERPRIORITY	varchar2	15	0
O_CLERK	varchar2	15	0
O_SHIPPRIOIRITY	number(p,s)	30	0
O_COMMENT	varchar2	79	0

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

You are a developer for an organization that 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 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.

Source Files

The source files are flat files that contain string data with blanks in the beginning and end. The source files for the Read transformation include Customer_FF and orders_FF.

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:

```
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
1, Smith John,Account Executive,DKR MANAGEMENT COMPANY INC,100 High
Street,5406,Anytown,TN,22342,USA,4047668150,2124031386,31/08/
1985,bwilliama@yahoo.com,    ACTIVE
2,Balasubramanian Krishna,Account Executive,EASTON & COMPANY,71 Congress
Parkway,789,Bangalore,Karnataka,38103,India,4046345228,4151689756,29/10/
1985,bmatthewc@univ.edu,    ACTIVE
```

```
3, Johnson Lars,Regional Sales Exec,GREATER BAY BANCORP,123 Snow St.,43543,St.
Paul,MN,55103,USA,4046581534,6122945948,7/9/1992,      ehpuniv.edu,INACTIVE
4,Zogby Kevin,Regional Sales Exec, HEWLETT-PACKARD,317 29th. St.,5856,San
Francisco,CA,94116,USA,4042662730,4155466814,7/8/1985,grobertwuniv.edu,    ACTIVE
5,Franklin Roosevelt,Sales Representative,JAYD TRADING,1511 Wacker
Dr,6334,Chicago,IL,60606,USA,7703965851,2065075486,20/10/
1982,trichard@univ.edu,INACTIVE
6,  Cruz Emilio,Sales Representative,JEFFERSON-PILOT LIFE INSURANCE,700 Ponce de
Leon
Blvd,757,Miami,FL,33134,USA,4043500799,2127655499,31/07/1983,ahelle@mailcity.com,
ACTIVE
7, King BB,Sales Representative,KUWAIT PETROLEUM CORPORATION,18 Beale
St,967,Memphis,TN,38103,USA,4046243979,2151717120,27/09/1989,
glizziem@univ.edu     ,INACTIVE
8,Presley Elvis,Sales Representative,PRINCIPIA PARTNERS,45 N Green
St.,43546,Tupelo,MS,38804,USA,4043733125,3311313591,26/07/1992,,    ACTIVE
9,Olson Floyd,Acct MGR., SOLITON ASSOCIATES INC.,21 Lake Harriet
Pkwy,869790,Mineaspolis,MN,55410,USA,7706425402,3232429056,27/08/1993,,INACTIVE
10,Chu Steven,Account Executive,WQXR,2100 Sepulveda Blvd,3434,Los
Angeles,CA,90049,USA,4042319005,2126509756,29/09/1988,akennetha@univ.edu,    ACTIVE
```

For example, the first and third rows have space in the beginning of the name:

```
1, Smith John,
3, Johnson Lars,
```

orders_FF columns and data

orders_FF contains the following columns:

```
OrderID
Customer_ID
Company
CompanyAddress
CompanyCity
CompanyState
CompanyZip
OrderContact
DeliveryAddress
DeliveryCity
DeliveryState
PaymentType
PaymentTerms
Title
DeliveryOption
DeliveryVendor
ConfirmationCode
OrderAmount
OrderType
ProductDescription
```

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,UPA,United Parcel Service Air,44162,$21.00 ,Generic,O/L/B P/W L/S TAWNY
SHIMMER .08 OZ.
O-6658,10110086,NRCA,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,O-L.B PW LIPSTYLO
LASTING PERFECTION .08 OZ.
O-8195,10110087,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,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,10019,GIORGIA TWITCHELL,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.
O-9352,10110089,YUASA TRADING COMPANY (AMERICA),150 EAST 52ND STREET,NEW
YORK,NY,10005,STEFFI 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.
O-9517,10110090,DAI ICHI KANGYO BANK,1 WORLD TRADE CENTRE  SUITE 49 - 11,NEW
YORK,NEW YORK,10048,AIKEN DOBRICK,1 WORLD TRADE CENTRE  SUITE 49 - 11,NEW YORK,NEW
YORK,American Express,CHARGE,Account Executive,UPR,United Parcel Service
Red,44167,$58.00 ,Generic,LASTING PERFECTION LIP COLOR HOLLYWOOD GLAMOUR 1.7 G MAUVE
ICE #752
O-9639,10110091,FIRST GLOBAL SECURITIES,614 EAST COLORADO BLVD.,PASADENA,CA,91101,
KIRSTENI SIPPEL,614 EAST COLORADO BLVD.,PASADENA,CA,American Express,CHARGE,Account
Executive,FSO,Federal Express Overnight,44168,$24.00 ,Generic,A/COL L PERFECTION L/S
REF P SUPREME .08 OZ.
O-9761,10110092,MILTON PARTNERS,56 MASON STREET,     GREENWICH     ,CT,6830,ORLANTA
DYSON,56 MASON STREET,GREENWICH,CT,American Express,CHARGE,Account
Executive,UPI,United Parcel Service International,44169,$75.20 ,Generic,LASTING
PERFECTION LIPSTYLO PEACH SU .08 OZ.
O-9883,10110093,   TAX ANALYSTS   ,6830 N. FAIRFAX DRIVE,ARLINGTON,VA,22213,NEWLIN
MCCART,6830 N. FAIRFAX DRIVE,ARLINGTON,VA,American Express,CHARGE,Account
Executive,FSO,Federal Express Overnight,44170,$275.40 ,Generic,O/L/B L PERFECTION L/
STYLO REF P SUPRE
O-5438,10110094,VECTORMEX,535 MADISON AVENUE,NEW YORK,NY,10022,LONNA HUGGINS,535
MADISON AVENUE,NEW YORK,NY,American Express,CHARGE,Account Executive,FSO,Federal
Express Overnight,44171,$60.00 ,Generic,LASTING PERFECTION DOUBLE PERFORMANCE
LIPSTICK PEACH SUPREME .08 OZ.
```

For example, the fourth row has space in the beginning and ending of the company name:

```
O-9130,10110088,   SCHRODER & COMPANY   ,
```

Target Files

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.

Dynamic Mapping

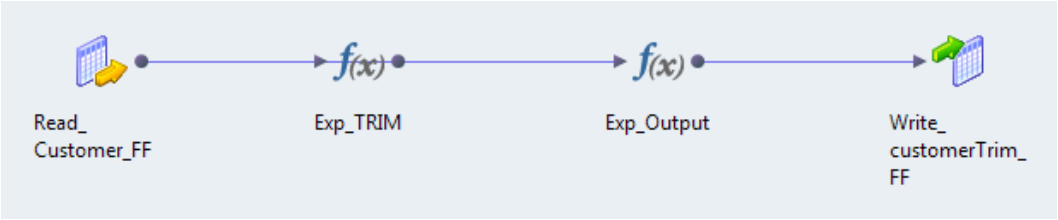
Create a mapping m_Replication_Template and configure the following dynamic mapping functionality:

- 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:

1. Reads the data from the appropriate source file based on the parameter value for the source data object.
2. Passes the new and changed columns to the downstream transformations through dynamic ports.
3. Expands the dynamic expression and processes the expression function for each generated port in the dynamic port.
4. Creates columns in the Write transformation based on the mapping flow and writes the transformed data to the appropriate target file based on the parameter value.

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.

Step 1. Configure the Read_Customer_FF Read Transformation

Configure the Read_Customer_FF Read transformation to use a parameter of type Resource to change the source data object between mapping runs.

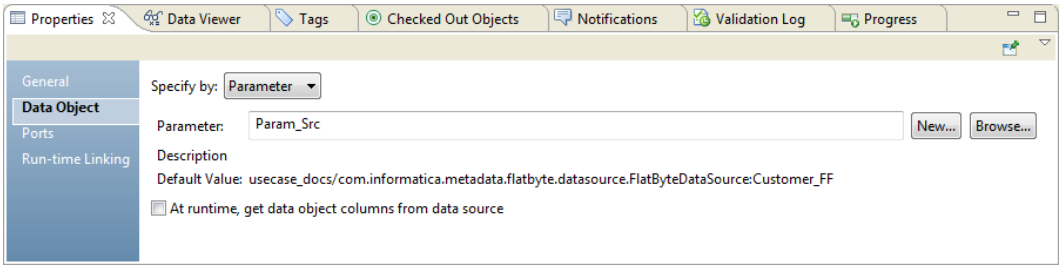
1. Add a Read transformation that represents the Customer_FF flat file data object.

   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_Dept/Customer_FF`, where `MRS` is the Model Repository Service and `Cust_Dept` 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:



Step 2. Configure the Exp_TRIM Expression Transformation

Add an Expression transformation Exp_TRIM to the mapping and configure the transformation to remove the spaces in the beginning and end of the strings.

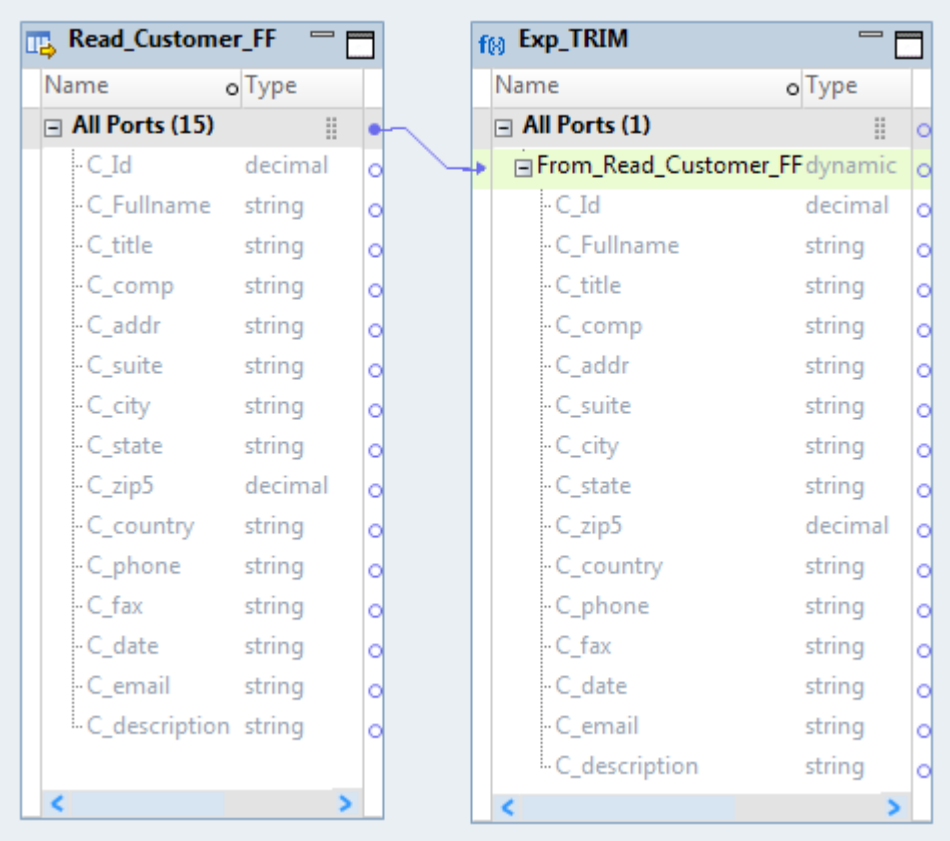
1. Create a dynamic port to receive columns from the Read transformation and define input rules to include only the string ports.
2. Create a dynamic output port and define a dynamic expression to remove the spaces in the beginning and end of the strings.

Create a Dynamic Port and Define Input Rules

Create a dynamic port to receive columns from the Read transformation. Define input rules to include only the string ports in the dynamic port.

1. 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:



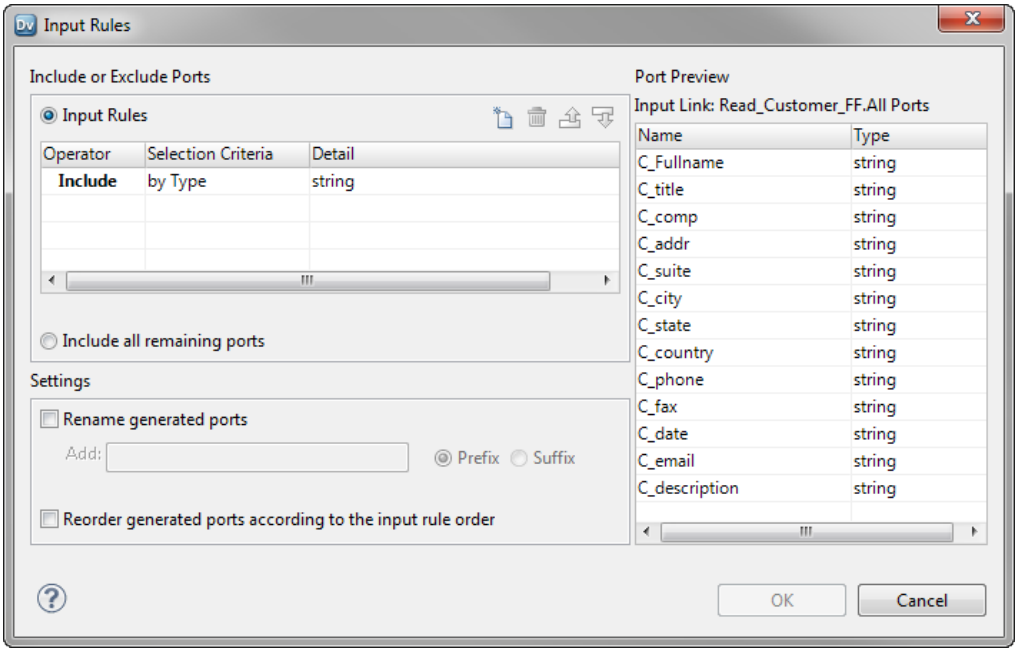
2. Right-click the dynamic port, and select **Edit Input Rules**.

   The **Input Rules** dialog box appears.

3. Select **by Type** from the **Selection Criteria** column.
4. Click the **Details** button to select the data type you want to include.
5. In the **Input Rule Detail: By Type** dialog box, select **string** data type from the list.

6. Verify the **Port Preview** area to ensure that only string ports appear.

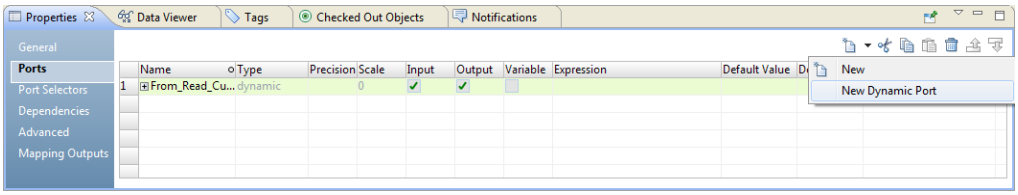
   The following image shows the updated input rule and the string ports in the **Port Preview** area of the **Input Rule** dialog box:



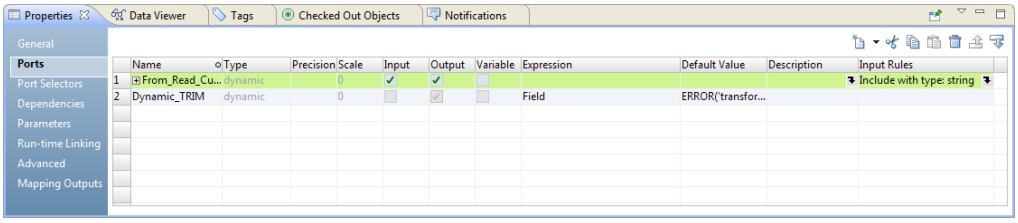
Create a Dynamic Port and Define a Dynamic Expression

Create a dynamic port as an output-only port in the Exp_TRIM transformation. Define a dynamic expression to remove the spaces in the beginning and end of the strings.

1. In the **Properties** view of the Exp_TRIM transformation, click the **Ports** tab.
2. Click **New Dynamic Port**.



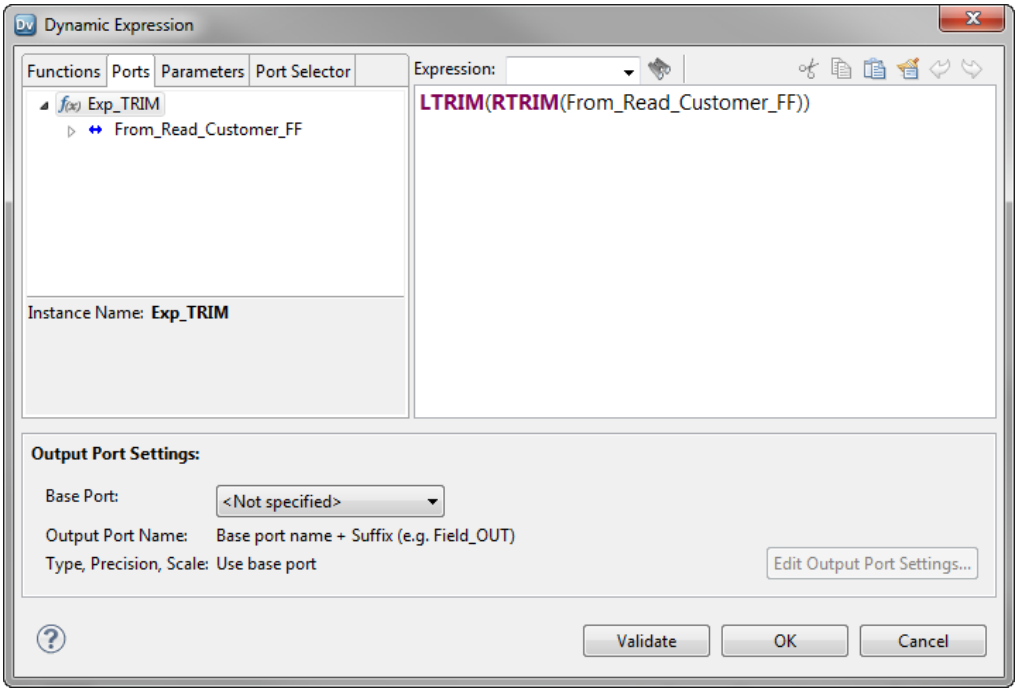
3. Clear the **Input** column to make this port an output-only port.
4. Rename the dynamic port that you created as `Dynamic_TRIM`.



5. In the **Expression** column for the `Dynamic_TRIM` dynamic port, click the **Open** button ( ).

   The **Dynamic Expression** window opens.

6. Replace the existing expression in the editor with the following expression:
   `LTRIM(RTRIM(From_Read_Customer_FF))`

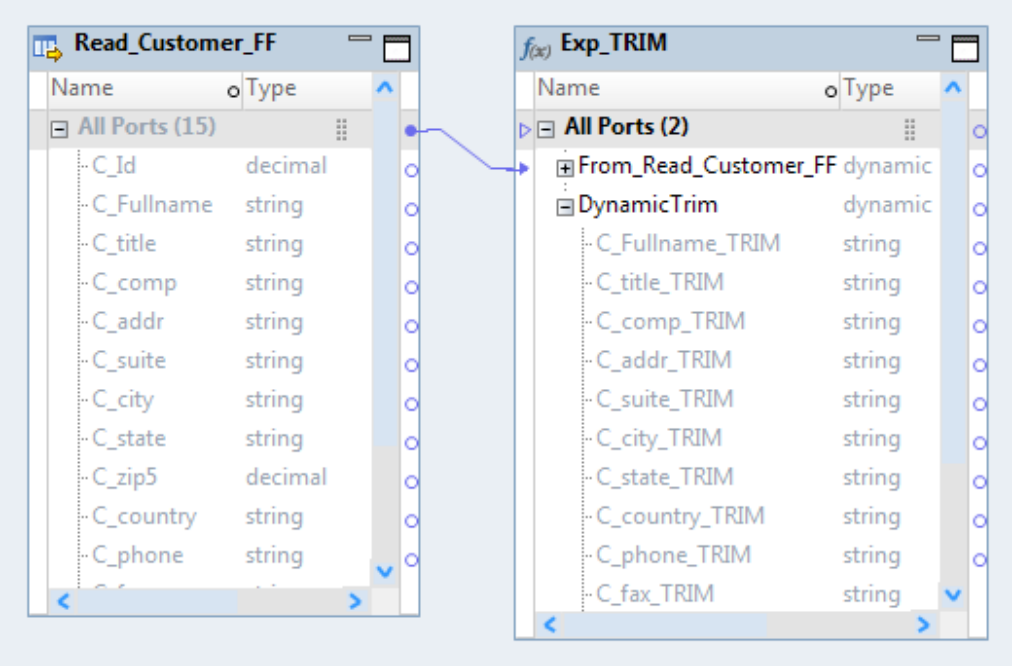


7. Click **Validate** to validate the expression.
8. Click **OK** to exit the **Validate Expression** dialog box.
9. Rename the output ports for the expression as follows:
   a. In the **Output Port Settings** area, select the Base Port as From_Read_Customer_FF.
   b. Click **Edit Output Port Settings**.

      The **Output Port Settings** dialog box appears.
   c. In the **Name** list, select **Base port name + Suffix**.
   d. In the **Suffix** box, enter `_TRIM`.
   e. Click **OK**.

10. Click **OK** to exit the **Dynamic Expression** editor.

    The following image shows the `Dynamic_TRIM` dynamic port with the renamed generated ports:



Step 3. Configure the Exp_Output Expression Transformation

Add an Expression transformation Exp_Output to the mapping. Create a dynamic port to get the output ports from the Exp_TRIM transformation. Create another dynamic port to get the ports from the Read transformation and define input rules to include only the unused ports.

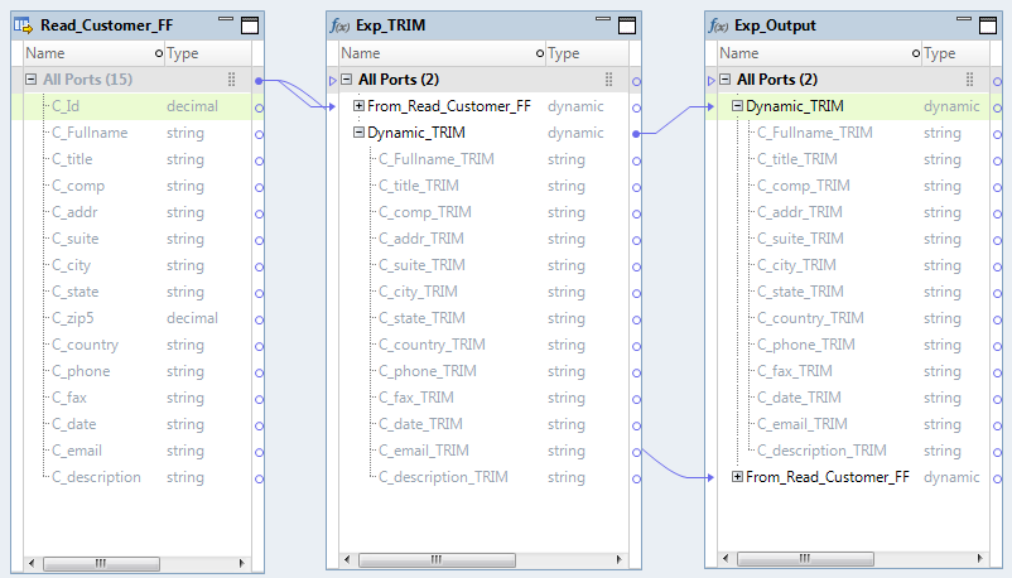
1. From the Exp_TRIM transformation, drag the `DynamicTrim` dynamic port to the All Ports group in the Exp_Output transformation.

   The Developer tool creates a dynamic port `DynamicTrim` in the Exp_Output transformation.

2. 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.

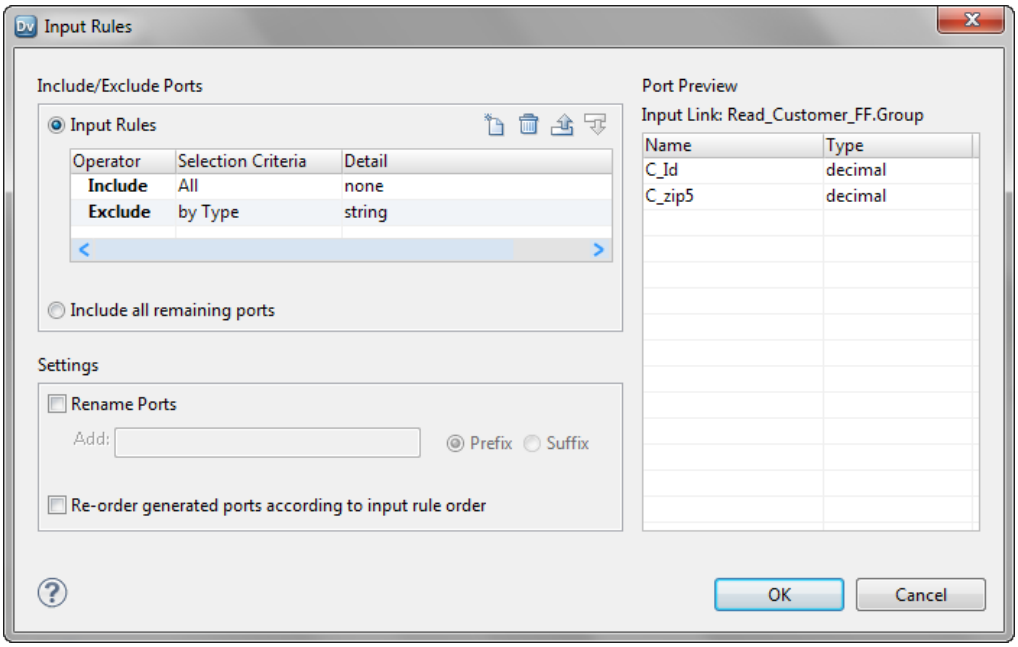


3. Right-click the From_Read_CUSTOMER_FF dynamic port, and select **Edit Input Rules**.

   The **Input Rules** dialog box appears.

4. Click the **New** icon to add an input rule.
5. Select **Exclude** in the **Operator** column.
6. Select **Type** in the **Selection Criteria** column.
7. Click the **Details** arrow to select the data type you want to include.
8. In the **Input Rule Detail: By Type** dialog box, select **string** data type from the list.
9. 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:



Step 4. Configure the Write_customerTrim_FF Write Transformation

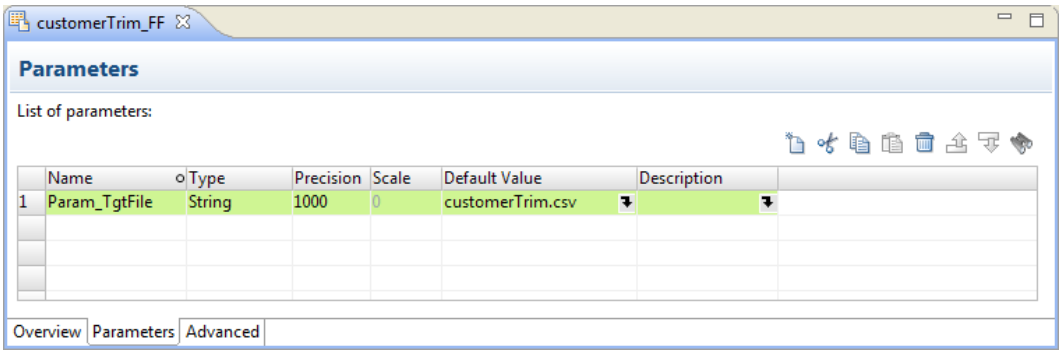
Create a customerTrim_FF data object and configure it to use a parameter of type String for the output file name. Configure the Write_customerTrim_FF transformation to create target files at run time based on the columns from the Exp_Output transformation.

Configure the Data Object to Use a Parameter

Create a customerTrim_FF data object to add as the Write transformation in the mapping. 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:

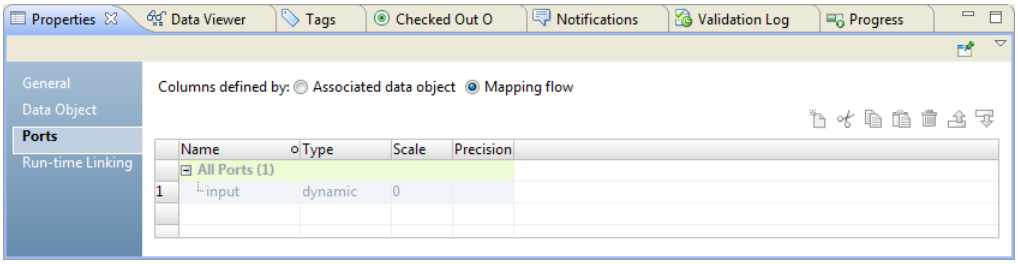


Create Target Columns from the Mapping Flow

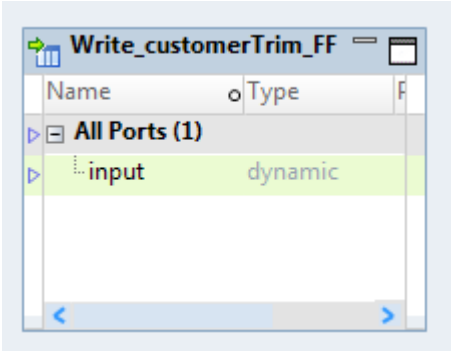
Add the Write transformation to the mapping and configure the Write_customerTrim_FF transformation to create target files at run time based on the columns from the Exp_Output transformation.

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:

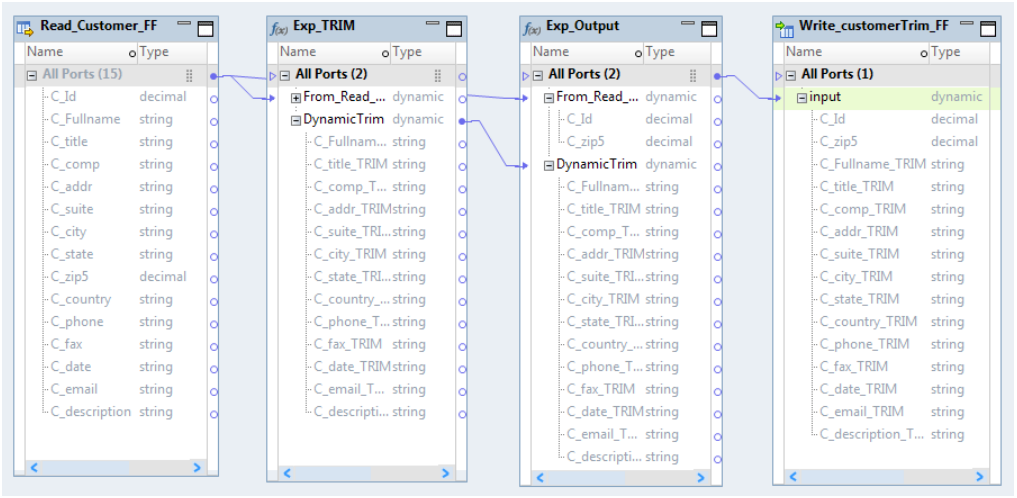


The following image shows the new dynamic port **input** in the Write_customerTrim_FF transformation:



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:



Step 5. Validate and Save 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.

Step 6. Run the Dynamic Mapping Against Different Sources and Targets

After you develop the dynamic mapping, you can run the mapping to access different sources and write to different targets based on the parameter values.

Run the Mapping for the Customer_FF Source

Run the `m_ReplicationTemplate` mapping with the default parameter values for the source data object and target file to view the result. The mapping reads from the Customer_FF source file and writes to the customerTrim.csv target file.

1. 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.

2. To view the results written to the target file, navigate to the target directory on the system where Informatica services is installed:

   ```
   <Informatica Installation Directory>\tomcat\bin\target
   ```

3. 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,bwilliama@yahoo.com,ACTIVE
   2,38103,Balasubramanian Krishna,Account Executive,EASTON & COMPANY,71 Congress
   Parkway,789,Bangalore,Karnataka,India,4046345228,4151689756,29/10/
   1985,bmatthewc@univ.edu,ACTIVE
   3,55103,Johnson Lars,Regional Sales Exec,GREATER BAY BANCORP,123 Snow St.,43543,St.
   Paul,MN,USA,4046581534,6122945948,7/9/1992,ehpuniv.edu,INACTIVE
   4,94116,Zogby Kevin,Regional Sales Exec,HEWLETT-PACKARD,317 29th. St.,5856,San
   Francisco,CA,USA,4042662730,4155466814,7/8/1985,grobertwuniv.edu,ACTIVE
   5,60606,Franklin Roosevelt,Sales Representative,JAYD TRADING,1511 Wacker
   Dr,6334,Chicago,IL,USA,7703965851,2065075486,20/10/1982,trichard@univ.edu,INACTIVE
   ```

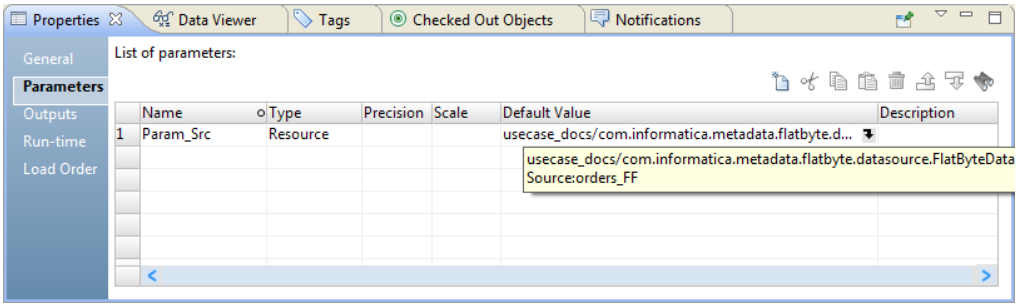
Change the Parameter Values

Change the parameter values for the source data object and the output file name of the target data object.

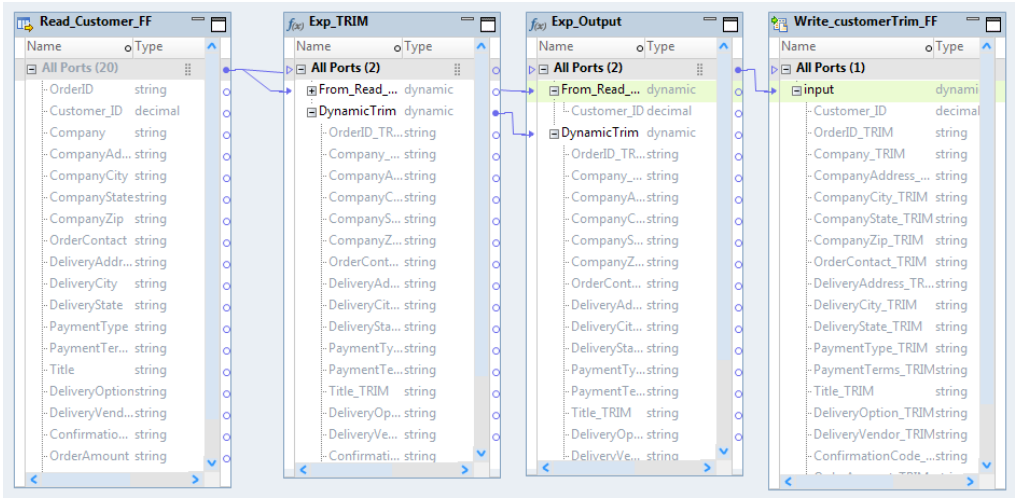
1. To change the parameter value for the source data object, perform the following steps:
   a. In the **Properties** view of the mapping, click the **Parameters** tab.
   b. Locate the `Param_Src` parameter for the source object.
   c. In the **Default Value** column, click the **Open** button ( ).

      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:



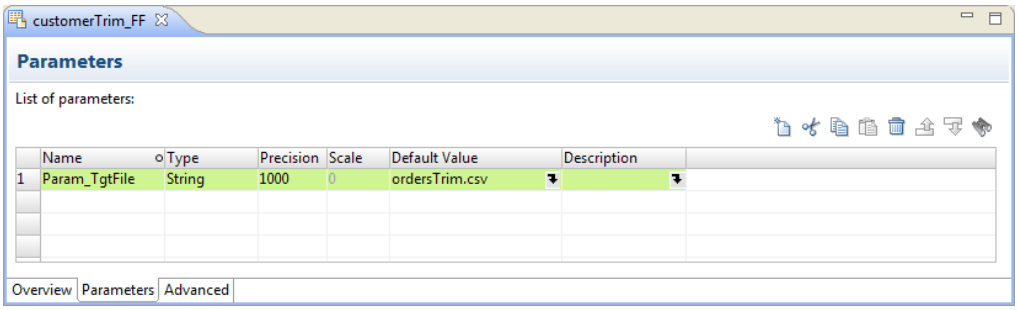
The following image shows the mapping that reflects the ports from the orders_FF data object for the Read transformation. The dynamic ports reflect new generated ports.



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:



Run the Mapping for the orders_FF Source

Validate the mapping and run the `m_ReplicationTemplate` mapping for a different source and target. The mapping reads from the orders_FF source file and writes to the ordersTrim.csv target file.

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. To view the results written to the target file, navigate to the target directory on the system where Informatica services is installed:

   ```
   <Informatica Installation Directory>\tomcat\bin\target
   ```
5. 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,O-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,O/L/B P/W L/S TAWNY
   SHIMMER .08 OZ.
   10110086,O-6658,NRCA,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,O-L.B PW LIPSTYLO LASTING PERFECTION .08 OZ.
   10110087,O-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,O/L/B P/W L/S TAWNY
   SHIMMER LASTING PERFECTION LIPSTYLO TAWNY SHIMMER .08 OZ.
   10110088,O-9130,SCHRODER & COMPANY,787 SEVENTH AVENUE,NEW YORK,NY,10019,GIORGIA
   TWITCHELL,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,O-9352,YUASA TRADING COMPANY (AMERICA),150 EAST 52ND STREET,NEW
   YORK,NY,10005,STEFFI 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.
   ```

CHAPTER 10

Mapping Administration

This chapter includes the following topics:

- Mapping Administration Overview, 206
- Viewing Properties for a Mapping Job, 207
- Viewing Summary Statistics for a Mapping Job, 207
- Viewing Detailed Statistics for a Mapping Job, 207
- Viewing Logs for a Mapping Job, 208
- Reissuing a Deployed Mapping Job, 208
- Canceling a Mapping Job, 209
- Reject Files, 209

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:

Task	Description
Enlarge a graph	Move the cursor over a graph, and then click the magnifying glass icon.
Enlarge a section of an enlarged graph	Drag the cursor to select an area to enlarge.
Switch between rows and bytes in the throughput graph	Click the Bytes option or the Rows option.
Choose which statistics are plotted on the throughput graph	In the throughput field, select the sources and targets that you want to view.

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:

Row Indicator	Meaning	Rejected By
0	Insert	Writer or target
1	Update	Writer or target
2	Delete	Writer or target
3	Reject. Marked for reject by an update strategy expression.	Writer
4	Rolled-back insert	Writer
5	Rolled-back update	Writer
6	Rolled-back delete	Writer
7	Committed insert	Writer
8	Committed update	Writer
9	Committed delete	Writer

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,O,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:

Column Indicator	Type of data	Writer Treats As
D	Valid data.	Good data. Writer passes it to the target database. The target accepts it unless a database error occurs, such as finding a duplicate key.
N	Null. The column contains a null value.	Good data. Writer passes it to the target, which rejects it if the target database does not accept null values.
T	Truncated. String data exceeded a specified precision for the column, so the value was truncated.	Bad data, if you configured the mapping target to reject overflow or truncated data.

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.

CHAPTER 11

Export to PowerCenter

This chapter includes the following topics:

- Export to PowerCenter Overview, 212
- PowerCenter Release Compatibility, 213
- Mapplet Export, 213
- Mappings with Parameters Export, 214
- Export to PowerCenter Options, 214
- Exporting an Object to PowerCenter, 215
- Export Restrictions, 216
- Rules and Guidelines for Exporting to PowerCenter, 218
- Troubleshooting Exporting to PowerCenter, 219

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.

Use the Developer tool or the required export command to export to PowerCenter to reuse the metadata.

To export data from the Model repository into the PowerCenter repository, complete the following tasks:

1. Export Model Repository objects to a file using the Developer tool or with the following command:
   `infacmd tools ExportObjects`

   Or, you can directly run `infacmd ipc ExportToPC` to export.

2. Convert the export file to a PowerCenter file using the following command:
   `infacmd ipc ExporttoPC`
3. Import the objects using the PowerCenter or with the following command:
   `pmrep importObjects`

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:

Option	Description
Project	Project in the Model repository from which to export objects.
Target release	PowerCenter release version.
Export selected objects to file	Exports objects to a PowerCenter XML file. If you select this option, specify the name and location of the export XML file.

Option	Description
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 a 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 the following objects to PowerCenter:

Objects that are not valid in PowerCenter

You cannot export a mapping or mapplet that contains any object that is not valid in PowerCenter, such as a Web Service Consumer transformation.

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 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

Mappings or mapplets that contain data warehouse sources or targets

You cannot export a mapping or mapplet to PowerCenter that contains Greenplum, Netezza, or Teradata PT objects.

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.

Verify the mapping type associated with the logical data object.

When you export a logical data object read mapping and a logical data object write mapping to PowerCenter, the read mapping appears under the mapplet in the Designer. The export process ignores the write mapping associated with the logical data object.

When you export a logical data object write mapping to PowerCenter, the export process fails.

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.

CHAPTER 12

Import From PowerCenter

This chapter includes the following topics:

- Import from PowerCenter Overview, 220
- Override Properties, 221
- Conflict Resolution, 223
- Import Summary, 224
- Data Type Conversion, 224
- Transformation Conversion, 224
- Parameter Conversion, 231
- PowerCenter Repository Connection Properties, 233
- Connection Assignments, 234
- Importing an Object from PowerCenter, 235
- Import Restrictions, 236
- Import Performance, 239

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.

Before you import objects from PowerCenter, run the infacmd ipc genReuseReportfromPC command in the command line to estimate the number of PowerCenter mappings that you can reuse in the Model repository in the native and Hadoop environments.

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.

You can import mappings from PowerCenter to reuse the metadata. Use the Developer tool or the required import command to import from PowerCenter.

To import data from PowerCenter into the Model Repository, complete the following tasks:

1. Export PowerCenter objects to a file using the PowerCenter Client or with the following command:
   `pmrep ExportObject`
2. Convert the export file to a Model repository file using the following command:
   `infacmd ipc importFromPC`
3. Import the objects using the Developer tool or with the following command:
   `infacmd tools importObjects`

You can import mappings with multiple pipelines, sessions, workflows, worklets from PowerCenter into the Model repository. Sessions within a workflow are imported as Mapping tasks in the Model repository. Workflows are imported as workflows within the Model repository. Worklets within a workflow are expanded and objects are imported into the Model repository.

After the import process completes, you can view the import summary.

Override Properties

When you import, the import process preserves the override, by default. You can choose to preserve or ignore the override properties of PowerCenter objects during the import process.

In PowerCenter, you can override some of the mapping properties. To override the properties, configure the properties for the session on the Mapping tab in the Workflow Manager.

With session overrides, you can override properties, such as Source File Name with different values in different sessions in PowerCenter. For example, you can set the **Source File Name** property with a value of 'OriginalFile'. Session 1 can override the value as "Session1_File" and Session 2 can override the value with "Session2_File".

When a PowerCenter mapping contains session overrides, the import process internally converts the session overrides by defining mapping or data object parameters in the Model repository and binds it to the parameter at the mapping task.

If you want to preserve the override properties in a Model repository, consider the following information:

- The import process creates reusable and 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.
- You can only preserve some override properties for the flat file sources, flat file targets, Lookup transformation, relational sources, and relational targets.

The following table lists the preserved override properties of PowerCenter objects in a Model repository:

PowerCenter Client Property	Developer Tool Property
Owner Name. Find in the relational source session properties.	**Owner**. Find in the mapping instance run-time properties.
Source Table Name. Find in the relational source session properties.	**Resource**. Find in the mapping instance run-time resource properties.
Reject file directory. Find in the relational target session properties.	**Reject file directory**. Find in the mapping instance run-time properties.
Reject filename. Find in the relational target session properties.	**Reject file name**. Find in the mapping instance run-time properties.
Target Table Name. Find in the relational target session properties.	**Target Table Name**. Find in the mapping instance run-time resource properties.
Pre SQL. Find in the relational source or target session properties.	**Pre SQL**. Find in the mapping instance advanced properties.
Post SQL. Find in the relational source or target session properties.	**Post SQL**. Find in the mapping instance advanced properties.
Sql Query. Find in the relational source or target session properties.	**SQL Query**. Find in the mapping instance query properties.
User Defined Join. Find in the source qualifier properties.	**Join**. Find in the mapping instance simple query properties.
Source Filter. Find in the source qualifier properties.	**Filter**. Find in the mapping instance simple query properties.
Lookup SQL override. Find in the relational Lookup transformation session properties.	**Lookup SQL override**. Find in the lookup query properties.
Lookup table name. Find in the relational Lookup transformation session properties.	**Resource**. Find in the lookup advanced run-time resource properties.
Lookup source filter. Find in the relational Lookup transformation session properties.	**Filter Query**. Find in the lookup query simple filter properties.
Lookup cache directory name. Find in the flat file or relational Lookup transformation session properties.	**Lookup cache directory name**. Find in the lookup run-time properties. For example, CacheDir (param).
Lookup source file name. Find in the flat file Lookup transformation session properties.	**Lookup source file name**. Find in the lookup data object specify by parameter properties.
Lookup source file directory. Find in the flat file Lookup transformation session properties.	**Lookup source file directory**. Find in the lookup data object specify by parameter properties.
Lookup column delimiter. Find in the flat file Lookup transformation session properties.	**Delimiters**. Find in the lookup data object advanced column delimiter properties.
Source File Directory. Find in the flat file source session properties.	**Source file directory**. Find in the data object advanced read properties.

PowerCenter Client Property	Developer Tool Property
Source Filename. Find in the flat file source session properties.	**Source file name**. Find in the data object advanced read properties.
Column Delimiter. Find in the flat file source session properties.	**Delimiters**. Find in the data object advanced column delimiters properties.
Reject filename. Find in the flat file target session properties.	**Reject file name**. Find in the mapping instance run-time properties.
Column Delimiter. Find in the flat file target session properties.	**Delimiters**. Find in the data object advanced column delimiters properties.
Merge File Directory. Find in the flat file target session properties.	**Merge file directory**. Find in the data object advanced write properties.
Merge File Name. Find in the flat file target session properties.	**Merge file name**. Find in the data object advanced write properties.
Output file directory. Find in the flat file target session properties.	**Output file directory**. Find in the data object advanced write properties.
Output filename. Find in the flat file target session properties.	**Output file name**. Find in the data object advanced write properties.
Reject file directory. Find in the flat file target session properties.	**Reject file directory**. Find in the mapping instance run-time properties.

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.

Data Type Conversion

Some PowerCenter data types are not valid in the Model repository. When you import PowerCenter objects with data types that are not valid, the import process converts them to valid, comparable data types in the Model repository.

You cannot convert session parameters, workflow parameters, and workflow variables in PowerCenter into the Model repository. The import process maps all the parameter or variable references in the session to a string representation in the corresponding mapping task in the Model repository.

The following table lists the PowerCenter repository data types that convert to corresponding Model repository data type in the import process:

PowerCenter Repository Data Type	Model Repository Data Type
Real	Double
Small Int	Integer
Nstring	String
Ntext	Text

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:

PowerCenter Transformation	Import Action
Aggregator	Imports with restrictions.
Data Masking	Fails to import.

PowerCenter Transformation	Import Action
External Procedure	Fails to import.
HTTP	Fails to import.
Identity Resolution	Fails to import.
Java	Imports with restrictions.
Joiner	Imports with restrictions.
Lookup	Imports with restrictions.
Normalizer	Imports with restrictions.
Rank	Imports with restrictions.
Sequence Generator	Imports with restrictions.
Sorter	Imports with restrictions.
Source Qualifier	Imports with restrictions. A source and Source Qualifier transformation import completely as one data object.
Stored Procedure	Fails to import.
Transaction Control	Fails to import.
SQL	Imports with restrictions.
Union	Imports with restrictions.
Unstructured Data	Fails to import.
XML Parser	Fails to import.
XML Generator	Fails to import.

Transformation Property Restrictions

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:

Transformation Property	Import Action
Transformation Scope	Ignore.

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:

Transformation Property	Import Action
Class Name	Ignore.
Function Identifier	Ignore.
Generate Transaction	Ignore.
Inputs Must Block	Ignore.
Is Partitionable	Ignore.
Language	Ignore.
Module Identifier	Ignore.
Output Is Deterministic	Ignore.
Output Is Repeatable	Ignore.
Requires Single Thread Per Partition	Ignore.
Runtime Location	Ignore.
Update Strategy Transformation	Ignore.

Joiner Transformation

The following table describes the import action for Joiner transformation properties:

Transformation Property	Import Action
Null Ordering in Master	Convert internally.
Null Ordering in Detail	Convert internally.
Transformation Scope	Convert internally.

Lookup Transformation

The following table describes the import action for Lookup transformation properties:

Transformation Property	Import Action
Cache File Name Prefix	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Lookup Cache Initialize	Ignore.
Lookup Cache Directory Name	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Lookup Caching Enabled	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Lookup Data Cache Size	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Lookup Index Cache Size	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Lookup Source is Static	Ignore.
Lookup Sql Override	Ignore if converted as a stand-alone transformation, and imports to Custom SQL Query when converted within a mapping.
Lookup Source Filter	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Pre-build Lookup Cache	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Re-cache from Lookup Source	Ignore if converted as a stand-alone transformation, and imports when converted within a mapping.
Recache if Stale	Ignore.
Subsecond Precision	Ignore.
Synchronize Dynamic Cache	Ignore.

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:

Transformation Property	Import Action
Reset	Ignore.
Restart	Ignore.

Rank Transformation

The following table describes the import action for Rank transformation properties:

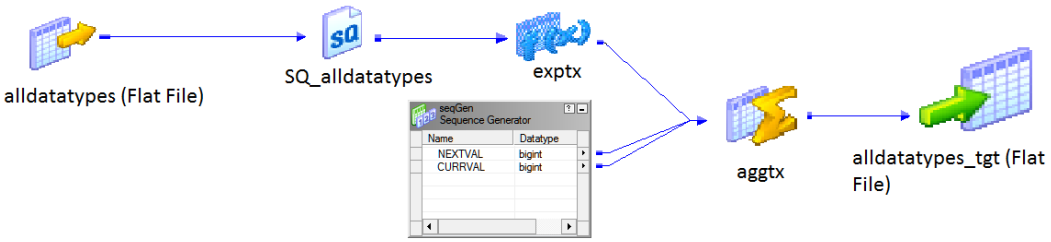
Transformation Property	Import Action
Transformation Scope	Ignore.

Sequence Generator Transformation

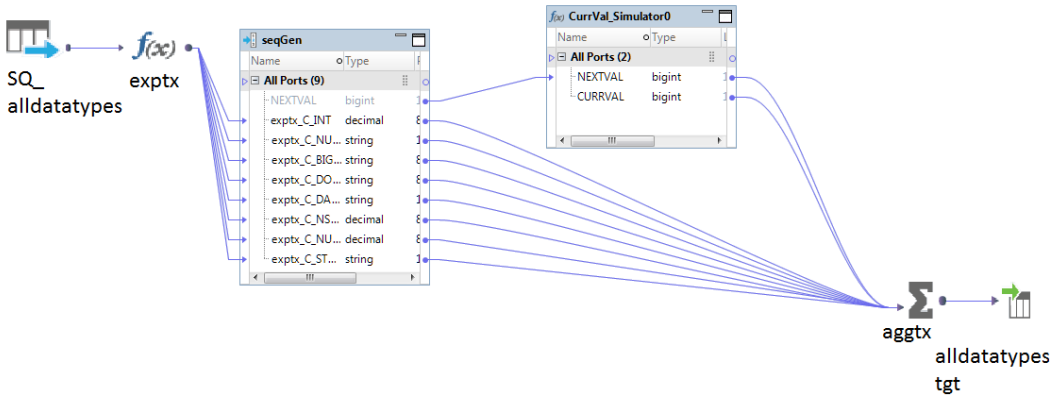
In PowerCenter, the Sequence Generator transformation has two ports, CURRVAL and NEXTVAL. In the Developer tool, the Sequence Generator transformation has only one port, NEXTVAL. When you import a Sequence Generator transformation in a mapping, the Developer tool creates an Expression transformation 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:

Transformation Property	Import Action
Current Value	Ignore.
Number of Cached Values	Ignore.

Sorter Transformation

The following table describes the import action for Sorter transformation properties:

Transformation Property	Import Action
Transformation Scope	Ignore.

Source Qualifier Transformation

The following table describes the import action for Source Qualifier transformation properties:

Transformation Property	Import Action
Number of Sorted Ports	Ignore.

SQL Transformation

The following table describes the import action for SQL transformation properties:

Transformation Property	Import Action
Auto Commit	Ignore.
Class Name	Ignore.
Connection Type	Fail import if set to dynamic connection object or full dynamic connect information.
Database Type	Fail import for Sybase, Informix, or Teradata.
Function Identifier	Ignore.
Generate Transaction	Ignore.
Inputs must Block	Ignore.
Is Partitionable	Ignore.
Language	Ignore.
Maximum Connection Pool	Ignore.
Module Identifier	Ignore.
Output is Deterministic	Ignore.
Output is Repeatable	Ignore.
Requires single Thread Per Partition	Ignore.
Runtime Location	Ignore.
SQL Mode	Fail import for script mode.
Transformation Scope	Ignore.
Treat DB Connection Failure as Fatal	Convert internally.
Update Strategy Transformation	Ignore.
Use Connection Pool	Ignore.

Union Transformation

The following table describes the import action for Union transformation properties:

Transformation Property	Import Action
Class Name	Ignore.
Function Identifier	Ignore.
Generate Transaction	Ignore.
Inputs Must Block	Ignore.
Is Partitionable	Ignore.
Language	Ignore.
Module Identifier	Ignore.
Output Is Deterministic	Ignore.
Output Is Repeatable	Ignore.
Requires Single Thread Per Partition	Ignore.
Runtime Location	Ignore.
Transformation Scope	Ignore.
Update Strategy Transformation	Ignore.

Parameter Conversion

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.

When you import from PowerCenter, consider the conversion information to import mappings with parameters, variables, or overrides.

Rules and Guidelines for Importing Mappings with Parameters

If you import a mapping with parameters, consider the following rules and guidelines:

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. The default value for a parameter with a String data type is a number sign (#). The default value for a parameter with a Numeric data type is 0.

The default value for a parameter with a Date/Time data type is 01/01/70 in the format mm/dd/yy. The default value for an SQL parameter is Empty.

PowerCenter mappings with the `IsExprVar` property are not valid 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.

Parameterized source owner in Netezza and Teradata objects or target table names in Teradata mappings fails to import.

When you import Netezza and Teradata objects that contains a parameterized source owner name property, the import process does not convert the property. The import process also ignores any parameterized target table names in Teradata mappings.

Rules and Guidelines for Importing Mapping with Variables

If you import a mapping with variables, consider the following rules and guidelines:

PowerCenter mappings with variables are not valid in the Model repository.

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.

Rules and Guidelines for Importing Overrides

If you import overrides, consider the following rules and guidelines:

Import process converts session overrides into corresponding parameter types.

When you import SQL-based overrides into the Model repository, the import process converts the overrides into a SQL parameter type. For example, when you import the PowerCenter mapping into the Model repository, the Pre SQL and Post SQL changes to an SQL parameter type. Also, a user-defined join and source filter converts to an SQL parameter type.

The remaining session override properties convert to String or to the corresponding parameter types. For example, Reject file converts to a String parameter type.

In PowerCenter, if you use a mapping parameter of the String type in session override properties, such as user defined joins, SQL query, or Post SQL, the import process brings in the referenced parameter as a String type. You should manually change the type of the mapping parameter assigned to the properties to SQL type after import.

System Parameter Conversion

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

Note: When a property is assigned a system parameter in the PowerCenter repository, the import process might not be able to convert the system parameters. The import process might set the default parameter value in the Model repository and you cannot edit the property value in the imported mapping.

PowerCenter Repository Connection Properties

When you import objects from a PowerCenter repository, you must specify the connection properties to the repository. The Developer tool uses the import properties to connect to the PowerCenter repository.

The following table describes the import properties:

Properties	Description
Host Name	PowerCenter domain gateway host name.
Port Number	PowerCenter domain gateway HTTP port number.
Release Number	PowerCenter release version.
Authentication Type	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. **Note:** Specify the user name and password if the authentication type is Native or LDAP.
User Name	PowerCenter repository user name.
Password	Password for the PowerCenter repository user name.

Properties	Description
Security Domain	If authentication type is LDAP, specify the LDAP security domain name. If not, enter Native.
Repository Name	PowerCenter repository name.
Code Page	Code page of the PowerCenter repository.

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.

If you import a PowerCenter mapping that contains a Source Qualifier with multiple downstream sources, the import process does not assign connections. You must manually assign connections to the objects.

When you edit the reusable Lookup transformation properties in the mapping level before importing it into the Model repository, the Developer tool imports a non-reusable Lookup transformation without assigning a connection. To resolve this issue, you can assign a connection manually to the Lookup relational data object or instance after import.

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

When you import objects from a PowerCenter repository to a Model repository, some restrictions and guidelines apply to the PowerCenter objects.

Certain restrictions apply to the following PowerCenter objects:

- Source and target
- Transformation
- Mapping
- Functions
- Mapping variables
- Session properties
- Workflows and Worklets
- Tasks

Source and Target Restrictions

The following source and target restrictions apply when you import PowerCenter objects:

- 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.
- You cannot import Salesforce sources or targets from PowerCenter.
- When you import Teradata sources from PowerCenter, the import process ignores the following properties:
  - Not Null property of a column
  - Output is Deterministic
  - Output is Repeatable
  - Number of Sorted Ports
  - Foreign Key
- When you import Teradata targets from PowerCenter, the import process ignores the following properties:
  - Not Null property of a column
  - Update Override
  - Target Table Prefix
  - Foreign Key

- When you import Netezza sources from PowerCenter, the import process ignores the following properties:
  - Foreign Key
  - Output is Deterministic
  - Output is Repeatable
  - Number of Sorted Ports
- When you import Netezza targets from PowerCenter, the import process ignores the following properties:
  - Foreign Key
  - Update Override
- When you import a Netezza source from PowerCenter, the import process imports the Netezza source with the same name under a newly created Netezza data object. However, when you create a read operation, the Source transformation within the read operation has a unique name appended with a number that is different from the imported source name.
- When you import a mapping with a Source Qualifier having an overridden SQL query, which selects a subset of the columns of the Source Qualifier, the Integration Service cannot run the mapping. To resolve this issue, remove the extra port and ensure that you match the SQL query to include all the columns in the source in the physical data object in the Developer tool.

Transformation Restrictions

The following transformation restrictions apply when you import PowerCenter objects:

- An expression in a transformation must contain 4,000 characters or fewer.
- The database type for an SQL 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.
- When you import a mapping with an empty Lookup cache directory name into the Model repository, the imported mapping is not valid. To resolve this issue, enter a valid value for the Lookup cache directory name under the run-time properties for the Lookup transformation in the Developer tool.

Mapping Restrictions

The following mapping restrictions apply when you import PowerCenter objects:

- When you import a mapping with supported objects or transformations, the import process creates a separate Model repository mapping for each pipeline in the PowerCenter mapping, preserving the target load order.
  The import process behavior remains the same for a workflow that contains a session that runs a mapping with multiple pipelines. Each mapping name is appended with a number that indicates the order in which mapping needs to run as mentioned in the target load order.
- If the mapping contains objects that are not supported in the Model repository, the pipelines are imported as a single mapping into the Model repository.
  The pipelines with the unsupported transformation or object are imported with broken links in the Model repository. In this case, the number of imported mappings can be less than the total number of pipelines in a PowerCenter mapping based on the target load order. If any pipeline in PowerCenter contains transformations or objects that are not supported in the Model repository, the pipeline is broken with the unconnected objects and it appears within a single mapping in the Model repository.

Functions Restrictions

When you import PowerCenter objects, you cannot import SetVariable and SetMax functions. Instead, you can convert the functions as mapping outputs in the Model repository.

Mapping Variables Restrictions

You cannot import mapping variables from PowerCenter into the Model repository.

Session Properties Restrictions

The following table lists the session properties that you can import from PowerCenter into the Model repository:

PowerCenter Session Property	Developer Tool Mapping Task Property
Session Log File Name	Mapping Task Log File Name
Session Log File directory	Mapping Task Log File Directory
Recovery Strategy > Fail task and continue workflow	Task Recovery Strategy > Skip
Recovery Strategy > Restart task are supported	Task Recovery Strategy > Restart
Java Classpath	Java Classpath
Enable high precision	High Precision
Session Sort Order	Sort Order
DateTime format string	Default Date Time Format
Save session log by > Save session log by timestamp	Mapping Task Log Save Type > Mapping task timestamp
Save session log by > Save session log by runs	Mapping Task Log Save Type > Mapping task run
Save session log for these runs	Save Mapping Task Logs For These Runs
Override tracing	Override Tracing Level
Enable HA recovery	Enable Recovery

The remaining session properties are not supported for import. If you import the unsupported properties, default values might appear.

Workflow Restrictions

The following workflow restrictions apply when you import PowerCenter objects:

- In PowerCenter, when you right-click a workflow link, you can set a link condition in the Workflow Manager. After you import the workflow in the Model repository, you can see the condition only as a comment, and you must manually convert the mapping based on the condition required.
- Workflow parameter, workflow variables, and session parameters are not supported for import from PowerCenter in the Model repository.

- You cannot import nested worklets from PowerCenter.

Task Restrictions

When you import PowerCenter objects, you can import Start, Session, Command, Worklet, and End tasks into the Model repository.

Import Performance

If you want to import mappings larger than 68 MB, import the mapping through command line for optimal performance. You can use the infacmd ipc ImportFromPC command to optimize import performance.

To improve performance during import from PowerCenter, you can add an option, -BlockSize to the infacmd ipc genReuseReportFromPC command.

Instead of importing every folder, you can import the required mappings from PowerCenter into the Model repository. Recommended Java heap size is 4 GB and blockSize is 100 for generating report especially if the user gets an out of memory error. You can also decrease the blockSize value in case of complex mappings.

CHAPTER 13

Performance Tuning

This chapter includes the following topics:

- Performance Tuning Overview, 240
- Optimization Methods, 241
- Optimizer Levels, 245
- Setting the Optimizer Level for a Developer Tool Mapping, 246
- Setting the Optimizer Level for a Deployed Mapping, 247

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 269.

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.

When you choose the normal or full optimizer level in the Developer tool, the early selection optimization method is enabled by default. When you use the auto optimizer level, the early selection optimization method is enabled in the following cases:

- The mapping runs in the native environment or on the Blaze engine.
- The mapping contains a data source that supports pushing filters to the source.

You can disable early selection if the optimization does not increase performance.

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.

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 simpliify 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.

Note: You cannot preview data in an optimized mapping.

Optimizer Levels

The Data Integration Service optimizes mappings based on the optimizer level that you configure. Configure the optimizer level property when you want the mapping to use an optimizer level other than the default.

You can choose one of the following optimizer levels:

Auto

The Data Integration Service applies optimizations based on the execution mode and mapping contents.

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.

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.

Default is Auto.

When you run the mapping from the **Run** menu or mapping editor in the Developer tool, 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 248

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 optimizer level you set in the mapping configuration. 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 14

Pushdown Optimization

This chapter includes the following topics:

- Pushdown Optimization Overview, 248
- Pushdown Types, 249
- Transformation Pushdown Logic, 251
- Pushdown Optimization to Sources, 252
- Pushdown Compatibility, 255
- Comparing the Output of the Data Integration Service and Sources, 267

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.

Related Topics:

- "Optimizer Levels" on page 245

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

- reorderAllowedForMultipleRowsWithSameKey
- reorderNotAllowedForRowsWithSameKey

The Data Integration Service can use full pushdown optimization for the following sources:

- Amazon Redshift
- Greenplum
- IBM DB2
- Microsoft SQL Server
- Netezza
- Oracle
- SAP HANA
- Snowflake
- Teradata

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, Microsoft SQL Server, Oracle, and ODBC sources such as Amazon Redshift, Greenplum, Netezza, Teradata, 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 Amazon Redshift, 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 Pushdown Logic

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 lookup match policy is not set to "Return All Rows."

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:

- Amazon Redshift
- 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:

Aggregate Functions	DB2-LUW	DB2i	DB2z/os
AVG	Yes	Yes	Yes
COUNT	Yes	Yes	Yes
FIRST	No	No	No
LAST	No	No	No
MAX	Yes	Yes	Yes

Aggregate Functions	DB2-LUW	DB2i	DB2z/os
MEDIAN	No	No	No
MIN	Yes	Yes	Yes
PERCENTILE	No	No	No
STDDEV	Yes	Yes	Yes
SUM	Yes	Yes	Yes
VARIANCE	Yes	Yes	Yes

The following table displays the aggregate functions that are valid in Amazon Redshift, Greenplum, Hive, MSSQL, Oracle, Sybase, and Teradata relational sources:

Aggregate Functions	Amazon Redshift	Greenplum	Hive	MSSQL	Oracle	Sybase	Teradata
AVG	Yes	Yes	Yes	Yes	Yes	Yes	Yes
COUNT	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FIRST	No	No	No	No	No	No	No
LAST	No	No	No	No	No	No	No
MAX	Yes	Yes	Yes	Yes	Yes	Yes	Yes
MEDIAN	No	No	No	No	Yes	No	No
MIN	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PERCENTILE	No	No	No	No	No	No	No
STDDEV	Yes	Yes	Yes	Yes	Yes	No	Yes
SUM	Yes	Yes	Yes	Yes	Yes	Yes	Yes
VARIANCE	Yes	Yes	Yes	Yes	Yes	No	Yes

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 driver 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:

- Amazon Redshift
- Greenplum
- Microsoft SQL Server
- Netezza
- SAP HANA
- Snowflake
- 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 Compatibility

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.

Operator Compatibility

When you use pushdown optimization, the Integration Service converts the expression in the transformation or workflow link by determining equivalent operators in the database. If there is no equivalent operator, the Integration Service processes the transformation logic.

You can push the following operators to the database using full pushdown optimization:

```
+ - * / % ||= > < >= <= <> != ^= not and or
```

Consider the following exceptions to operators:

- Data Integration Service can push these operators to the following sources only when they are included in Filter transformation logic:
  - DB2 for i5/OS or z/OS
  - Non-relational
  - SAP
- You cannot push the `/ % || <> ^= not` operator to Non-relational sources.

  Note: Nonrelational sources are IMS, VSAM, and sequential data sets on z/OS.
- You cannot push down `+ - * / % || not` operators to SAP sources.
- You cannot push down `% ||` operators to ODBC sources.

Functions for IBM DB2

Informatica functions are not available for non-relational sources on z/OS. The following table displays the Informatica functions available for pushdown optimization for IBM DB2 sources:

Function	DB2 for i5/OS [1]	DB2 for LUW	DB2 for z/OS [1]
ABS()	No	Yes	No
ADD_TO_DATE()	Yes	Yes	Yes
ASCII()	Yes	Yes	Yes
AVG()	Yes	Yes	Yes
CEIL()	Yes	Yes	Yes
CHR()	No	Yes	No
CHRCODE()	No	Yes	Yes
CONCAT()	Yes	Yes	Yes
COS()	Yes	Yes	Yes
COSH()	Yes	Yes	Yes
COUNT()	Yes	Yes	Yes
DATE_COMPARE()	Yes	Yes	Yes
DECODE()	No	Yes	No
EXP()	No	Yes	No
FLOOR()	No	Yes	No

Function	DB2 for i5/OS[1]	DB2 for LUW	DB2 for z/OS[1]
GET_DATE_PART()	Yes	Yes	Yes
IIF()	No	Yes	No
IN()	No	Yes	No
INSTR()	Yes	Yes	Yes
ISNULL()	Yes	Yes	Yes
LENGTH()	Yes	Yes	Yes
LN()	Yes	Yes	Yes
LOG()	Yes	Yes	Yes
LOWER()	Yes	Yes	Yes
LTRIM()	Yes	Yes	Yes
MAX()	Yes	Yes	Yes
MIN()	Yes	Yes	Yes
MOD()	Yes	Yes	Yes
POWER()	Yes	Yes	Yes
ROUND(DATE)	No	No	Yes
ROUND(NUMBER)	Yes	Yes	Yes
RTRIM()	Yes	Yes	Yes
SIGN()	Yes	Yes	Yes
SIN()	Yes	Yes	Yes
SINH()	Yes	Yes	Yes
SOUNDEX()	No	Yes[1]	No
SQRT()	No	Yes	No
STDDEV()	Yes	Yes	Yes
SUBSTR()	Yes	Yes	Yes
SUM()	Yes	Yes	Yes
SYSTIMESTAMP()	Yes	Yes	Yes
TAN()	Yes	Yes	Yes

Function	DB2 for i5/OS [1]	DB2 for LUW	DB2 for z/OS [1]
TANH()	Yes	Yes	Yes
TO_BIGINT	Yes	Yes	Yes
TO_CHAR(DATE)	Yes	Yes	Yes
TO_CHAR(NUMBER)	Yes	Yes [2]	Yes
TO_DATE()	Yes	Yes	Yes
TO_DECIMAL()	Yes	Yes [3]	Yes
TO_FLOAT()	Yes	Yes	Yes
TO_INTEGER()	Yes	Yes	Yes
TRUNC(NUMBER)	Yes	Yes	Yes
UPPER()	Yes	Yes	Yes
VARIANCE()	Yes	Yes	Yes

. [1]The Data Integration Service can push these functions to the source only when they are included in Filter transformation logic.
. [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.

If a function is not listed, the Integration Service cannot push that function to any database.

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.

Functions for Cloud Data Warehouse Applications

The following table displays the Informatica functions available for pushdown optimization for Amazon Redshift and Snowflake:

Function	Amazon Redshift	Snowflake
ABS()	Yes	Yes
ADD_TO_DATE()	Yes	Yes
ASCII()	No	Yes
AVG()	Yes	Yes
CEIL()	Yes	Yes
CHR()	Yes	Yes
CHRCODE()	No	Yes
CONCAT()	Yes	Yes
COS()	Yes	No
COSH()	No	Yes
COUNT()	Yes	No
DATE_COMPARE()	Yes	No
DATE_DIFF()	Yes	Yes
DECODE()	Yes	Yes
EXP()	Yes	Yes
FIRST()	No	Yes
FLOOR()	Yes	Yes
GET_DATE_PART()	Yes	Yes
IIF()	Yes	Yes
IN()	Yes	No
INITCAP()	Yes	Yes
INSTR()	Yes	No
ISNULL()	Yes	No
LAST_DAY()	Yes	No
LENGTH()	Yes	Yes

Function	Amazon Redshift	Snowflake
LN()	Yes	Yes
LOG()	No	Yes
LOWER()	Yes	Yes
LPAD()	Yes	Yes
LTRIM()	Yes	Yes
MAX()	Yes	Yes
MEDIAN()	No	Yes
MIN()	Yes	Yes
MOD()	Yes	Yes
POWER()	Yes	Yes
ROUND(DATE)	No	Yes
ROUND(NUMBER)	Yes	Yes
RPAD()	Yes	Yes
RTRIM()	Yes	Yes
SIGN()	Yes	Yes
SIN()	Yes	No
SINH()	No	Yes
SQRT()	Yes	Yes
STDDEV()	Yes	Yes
SUBSTR()	Yes	Yes
SUM()	Yes	No
SYSTIMESTAMP()	Yes	No
TAN()	Yes	No
TANH()	No	Yes
TO_BIGINT	Yes	Yes
TO_CHAR(DATE)	Yes	Yes
TO_CHAR(NUMBER)	Yes	Yes

Function	Amazon Redshift	Snowflake
TO_DATE()	Yes	Yes
TO_DECIMAL()	Yes	Yes
TO_DECIMAL38()	Yes	Yes
TO_FLOAT()	Yes	Yes
TO_INTEGER()	Yes	Yes
TRUNC(DATE)	Yes	Yes
TRUNC(NUMBER)	Yes	Yes
UPPER()	Yes	Yes
VARIANCE()	Yes	Yes

If a function is not listed, the Integration Service cannot push that function to any database.

Amazon Redshift Function Exceptions

The Data Integration Service cannot push supported functions to Amazon Redshift databases under certain conditions.

- To use pushdown optimization, you must set the value of the **Multiple Matches** Lookup transformation property to **Return all rows**.
- To push TRUNC(DATE) to Amazon Redshift, you must define the date and format arguments. Otherwise, the agent does not push the function to Amazon Redshift.
- The aggregator functions for Amazon Redshift accept only one argument, a field set for the aggregator function. The filter condition argument is ignored. In addition, verify that all fields mapped to the target are listed in the GROUP BY clause.
- To push TO_DATE() to Amazon Redshift, you must define the string and format arguments.
- To push TO_CHAR() to Amazon Redshift, you must define the date and format arguments.
- Do not specify a format for SYSTIMESTAMP() to push the SYSTIMESTAMP to Amazon Redshift. The Amazon Redshift database returns the complete time stamp.
- To push INSTR() to Amazon Redshift, you must only define string, search_value, and start arguments. Amazon Redshift does not support occurrence and comparison_type arguments.
- The flag argument is ignored when you push TO_BIGINT and TO_INTEGER to Amazon Redshift.
- The CaseFlag argument is ignored when you push IN() to Amazon Redshift.
- If you use the NS format as part of the ADD_TO_DATE() function, the agent does not push the function to Amazon Redshift.
- If you use any of the following formats as part of the TO_CHAR() and TO_DATE() functions, the agent does not push the function to Amazon Redshift:

  - NS

  - SSSS

- SSSSS
- RR

- To push TRUNC(DATE), GET_DATE_PART(), and DATE_DIFF() to Amazon Redshift, you must use the following formats:
  - D
  - DDD
  - HH24
  - MI
  - MM
  - MS
  - SS
  - US
  - YYYY

Functions for Data Warehouse Applications

The following table displays the Informatica functions available for pushdown optimization for Greenplum, Netezza, and Teradata sources:

Function	Greenplum	Netezza	Teradata
ABS()	Yes	Yes	Yes
ADD_TO_DATE()	Yes	Yes	Yes
ASCII()	Yes	Yes	No
AVG()	Yes	Yes	Yes
CEIL()	Yes	Yes	Yes
CHR()	Yes	Yes	No
CHRCODE()	Yes	Yes	No
CONCAT()	Yes	Yes	Yes
COS()	Yes	Yes	Yes
COSH()	Yes	Yes	Yes
COUNT()	Yes	Yes	Yes
DATE_COMPARE()	Yes	Yes	Yes
DECODE()	Yes	Yes	Yes
EXP()	Yes	Yes	Yes
FLOOR()	Yes	Yes	Yes

Function	Greenplum	Netezza	Teradata
GET_DATE_PART()	Yes	Yes	Yes
IIF()	Yes	Yes	Yes
IN()	No	No	Yes
INITCAP()	Yes	Yes	No
INSTR()	No	Yes	Yes
ISNULL()	No	No	Yes
LAST_DAY()	No	Yes	No
LENGTH()	Yes	Yes	Yes
LN()	Yes	No	Yes
LOG()	Yes	Yes	Yes
LOWER()	Yes	Yes	Yes
LPAD()	Yes	Yes	No
LTRIM()	Yes	Yes	Yes
MAX()	Yes	Yes	Yes
MIN()	Yes	Yes	Yes
MOD()	Yes	Yes	Yes
POWER()	Yes	Yes	Yes
ROUND(NUMBER)	Yes	Yes	Yes
RPAD()	Yes	Yes	No
RTRIM()	Yes	Yes	Yes
SIGN()	Yes	Yes	Yes
SIN()	Yes	Yes	Yes
SINH()	Yes	Yes	Yes
SQRT()	Yes	Yes	Yes
STDDEV()	Yes	Yes	Yes
SUBSTR()	Yes	Yes	Yes
SUM()	Yes	Yes	Yes

Function	Greenplum	Netezza	Teradata
SYSTIMESTAMP()	Yes	Yes	No
TAN()	Yes	Yes	Yes
TANH()	Yes	Yes	Yes
TO_BIGINT	Yes	Yes	Yes
TO_CHAR(DATE)	Yes	Yes	Yes
TO_CHAR(NUMBER)	Yes	Yes	Yes
TO_DATE()	Yes	Yes	Yes
TO_DECIMAL()	Yes	Yes	Yes
TO_DECIMAL38()	Yes	Yes	Yes
TO_FLOAT()	Yes	Yes	Yes
TO_INTEGER()	Yes	Yes	Yes
TRUNC(DATE)	Yes	Yes	No
TRUNC(NUMBER)	Yes	Yes	Yes
UPPER()	Yes	Yes	Yes
VARIANCE()	Yes	Yes	Yes

If a function is not listed, the Integration Service cannot push that function to any database.

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.

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.

Functions for Enterprise Applications

The following table displays the Informatica functions available for pushdown optimization for Hive, SAP and SAP HANA sources:

Function	Hive	SAP [1]	SAP HANA
ABS()	Yes	No	Yes
AVG()	Yes	No	Yes
CEIL()	Yes	No	Yes
CHR()	No	No	Yes
CHRCODE()	No	No	Yes
CONCAT()	Yes	No	Yes
COS()	Yes	No	Yes
COSH()	No	No	Yes
COUNT()	Yes	No	Yes
DATE_COMPARE()	No	No	Yes
DATE_DIFF()	No	No	Yes
DECODE()	Yes	No	Yes
EXP()	Yes	No	No
FLOOR()	Yes	No	Yes
GET_DATE_PART()	No	No	Yes
IIF()	Yes	No	Yes
ISNULL()	Yes	No	Yes
LAST_DAY()	No	No	Yes
LENGTH()	Yes	No	Yes
LN()	Yes	No	Yes
LOG()	Yes	No	Yes
LOWER()	Yes	No	Yes

Function	Hive	SAP [1]	SAP HANA
LPAD()	Yes	No	Yes
LTRIM()	Yes	No	Yes
MAX()	Yes	No	Yes
MIN()	Yes	No	Yes
MOD()	Yes	No	Yes
POWER()	Yes	No	Yes
ROUND(NUMBER)	Yes	No	Yes
RPAD()	Yes	No	Yes
RTRIM()	Yes	No	Yes
SIGN()	No	No	Yes
SIN()	Yes	No	Yes
SINH()	No	No	Yes
SQRT()	Yes	No	Yes
STDDEV()	No	No	Yes
SUBSTR()	Yes	No	Yes
SUM()	Yes	No	Yes
SYSTIMESTAMP()	No	No	Yes [2]
TAN()	No	No	Yes
TO_BIGINT	Yes	No	Yes
TO_CHAR(DATE)	No	No	Yes
TO_CHAR(NUMBER)	No	No	Yes
TO_DATE()	No	Yes	Yes
TO_DECIMAL()	No	No	Yes
TO_DECIMAL38()	Yes	Yes	Yes
TO_FLOAT()	No	No	Yes
TO_INTEGER()	Yes	No	Yes
TRUNC(DATE)	No	No	Yes

Function	Hive	SAP [1]	SAP HANA
TRUNC(NUMBER)	No	No	Yes
UPPER()	Yes	No	Yes

1. The Data Integration Service can push these functions to the source only when they are included in Filter transformation logic.
2. SYSTIMESTAMP() supports only the SS argument.

If a function is not listed, the Integration Service cannot push that function to any database.

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 process the transformation logic for Hive sources when you use the following functions with the date datatype:

- CONCAT
- MAX
- MIN
- ROUND
- TO_BIGINIT
- TO_INTEGER

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: IIF(col_varchar2 = 'CA', TRUE, FALSE). You need the database to return rows that match 'CA.' However, if you push this transformation logic to a database that is not case sensitive, it returns rows that match the values 'Ca,' 'ca,' 'cA,' and 'CA.'

Numeric values converted to character values

The Data Integration Service and a database can convert the same numeric value to a character value in different formats. The database might convert numeric values to an unacceptable character format. For example, a table contains the number 1234567890. When the Data Integration Service converts the number to a character value, it inserts the characters '1234567890.' However, a database might convert the number to '1.2E9.' The two sets of characters represent the same value.

Date formats for TO_CHAR and TO_DATE functions

The Data Integration Service uses the date format in the TO_CHAR or TO_DATE function when the Data Integration Service pushes the function to the database. Use the TO_DATE functions to compare date or time values. When you use TO_CHAR to compare date or time values, the database can add a space or leading zero to values such as a single-digit month, single-digit day, or single-digit hour. The database comparison results can be different from the results of the Data Integration Service when the database adds a space or a leading zero.

Precision

The Data Integration Service and a database can have different precision for particular datatypes. Transformation datatypes use a default numeric precision that can vary from the native datatypes. The results can vary if the database uses a different precision than the Data Integration Service.

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.

CHAPTER 15

Partitioned Mappings

This chapter includes the following topics:

- Partitioned Mappings Overview, 269
- One Thread for Each Pipeline Stage, 270
- Multiple Threads for Each Pipeline Stage, 271
- Partitioned Flat File Sources, 273
- Partitioned Relational Sources, 274
- Partitioned Flat File Targets, 276
- Partitioned Relational Targets, 280
- Partitioned Transformations, 281
- Maintain Order in a Partitioned Mapping, 284
- Override the Maximum Parallelism for a Mapping, 285
- Troubleshooting Partitioned Mappings, 289

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.

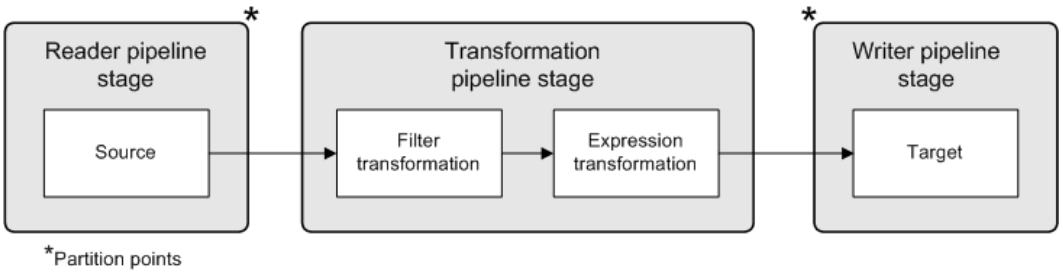
Each mapping contains one or more pipelines. A pipeline consists of a Read transformation and all the transformations that receive data from that Read transformation. The Data Integration Service separates a mapping pipeline into pipeline stages and then performs the extract, transformation, and load 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:



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:

Reader Thread	Transformation Thread	Writer Thread
Row Set 1	-	-
Row Set 2	Row Set 1	-
Row Set 3	Row Set 2	Row Set 1
Row Set 4	Row Set 3	Row Set 2
Row Set n	Row Set (n-1)	Row Set (n-2)

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:

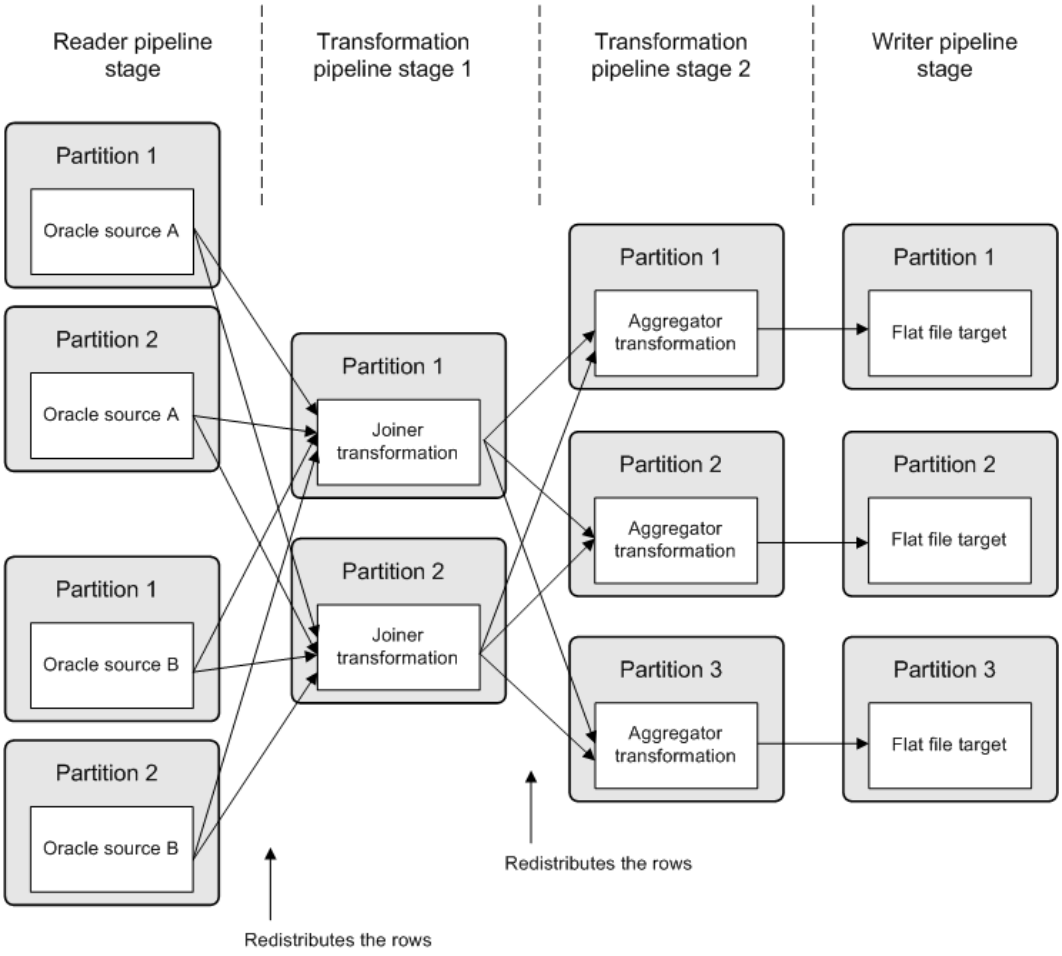
Divides the data into partitions.

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:

Partition	Rows Read
Partition #1	1,3,5,8,9
Partition #2	2,4,6,7,10

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:

Partition	Rows Read
Partition #1	1,2,3,4,5
Partition #2	6,7,8,9,10

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 simply 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:

```
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:

```
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:

```
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_a name>
UNION ALL
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_b name>
```

```
UNION ALL
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition4_a name>
UNION ALL
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition4_b name>
UNION ALL
```

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 Services 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.

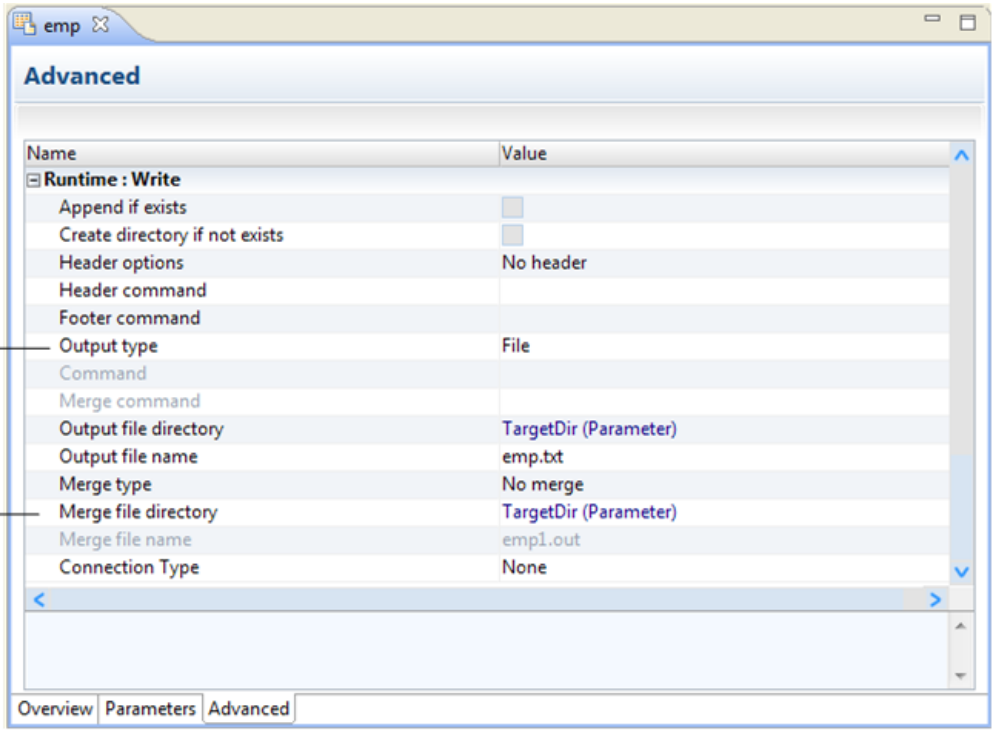
You can enter a different value to configure multiple output file directories specific to the flat file data object. Enter multiple directories separated by semicolons for the property or for the user-defined parameter assigned to the property.

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.

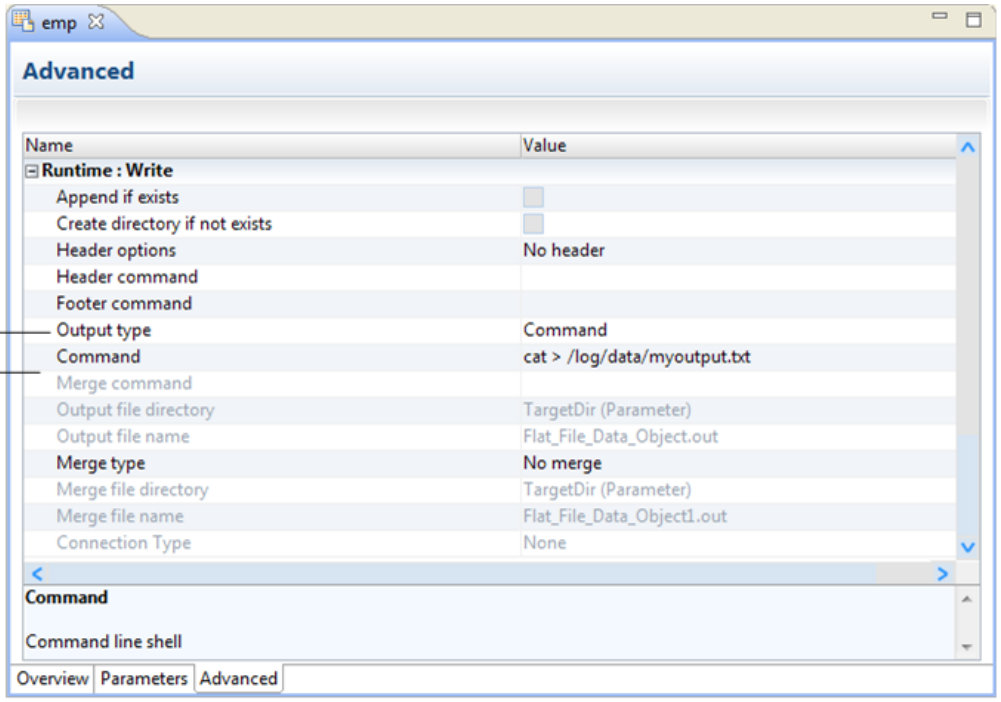
If you configure the flat file data object to merge target data and the Data Integration Service does not create partitions for the target, the Data Integration Service ignores the merge options. The service writes to the file defined in the **Output File Name** and **Output File Directory** properties.

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/`

`myoutput.txt`, 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 a pass-through 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 equality 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,ItemD,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,ItemD,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. If you set the maximum parallelism value for the mapping to Auto after defining a suggested parallelism value for a transformation, the Data Integration Service ignores the suggested parallelism value.

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:

Suggested Parallelism Value	Description
1	The Data Integration Service uses one thread to run the transformation.
Auto	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.
Greater than 1	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.

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:

Execution Instances Value	Description
1	The Data Integration Service uses one thread to run the transformation. Default for the Address Validator transformation.
Auto	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.
Greater than 1	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.

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.

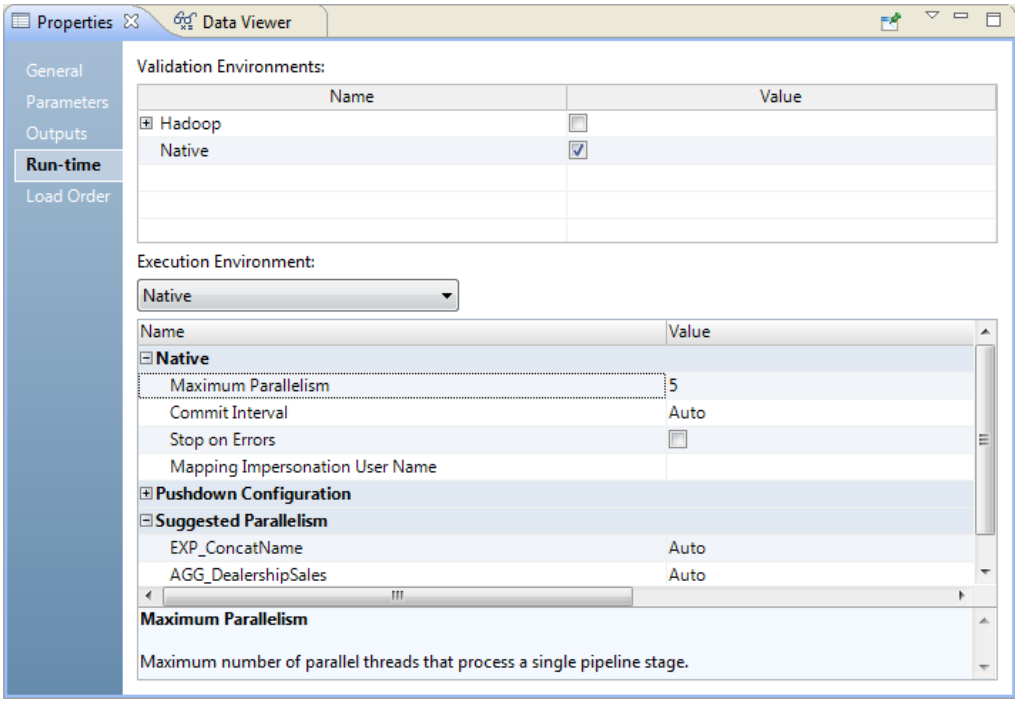
Overriding the Maximum Parallelism Value

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.
3. Select **Native** for the **Execution Environment**.
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.
```

CHAPTER 16

Developer Tool Naming Conventions

This chapter includes the following topics:

- Transformation Naming Conventions, 290
- Object Type Naming Conventions, 292
- Workflow Object Naming Conventions, 292

Transformation Naming Conventions

The following table contains standardized naming conventions for transformation types:

Transformation Type	Suggested Naming Convention
Association	AST_<transformation name>
Address Validator	AGG_<transformation name>
Aggregator	AGG_<transformation name>
Association	AST_<transformation name>
Bad Record Exception	EXC_<transformation name>
Case Converter	CCO_<transformation name>
Classifier	CLA_<transformation name>
Comparison	CMP_<transformation name>
Consolidation	CNS_<transformation name>
Data Masking	DMK_<transformation name>
Data Processor	DPR_<transformation name>
Decision	DEC_<transformation name>

Transformation Type	Suggested Naming Convention
Duplicate Record Exception	EXC_<transformation name>
Expression	EXP_<transformation name>
Filter	FIL_<transformation name>
Java	JTX_<transformation name>
Joiner	JNR_<transformation name>
Key Generator	KGN_<transformation name>
Labeler	LAB_<transformation name>
Lookup	LKP_<transformation name>
Match	MAT_<transformation name>
Merge	MRG_<transformation name>
Normalizer	NRM_<transformation name>
Parser	PRS_<transformation name>
Rank	RNK_<transformation name>
Read	SRC_<transformation name>
REST Provider	RESTP_<transformation name>
REST Web Service Consumer	RESTWS_<transformation name>
Router	RTR_<transformation name>
Sequence Generator	SEQ_<transformation name>
Sorter	SRT_<transformation name>
SQL	SQL_<transformation name>
Standardizer	STD_<transformation name>
Union	UN_<transformation name>
Update Strategy	UPD_<transformation name>
Web Service Consumer	WSC_<transformation name>
Weighted Average	WAV_<transformation name>
Write	WRT_<update type>_<target name>

Object Type Naming Conventions

The following table contains standardized naming conventions for repository objects:

Developer Tool Object	Suggested Naming Convention
Application	APP_<application name>
Customized data object	CDO_<data object name>
Logical data object	LDO_<data object name>
Logical data object model	LDOM_<model name>
Mapping	M_<process>_<source system>_<target name>
Mapplet	MPLT_<mapplet name>
Physical data object	PDO_<data object name>
Profile	PRFL_<profile name>
Rule	Rule_<rule name>
Scorecard	SCD_<scorecard name>
SQL data service	SDS_<data service name>
Target	T_<target name>
Virtual schema	VS_<schema name>
Virtual stored procedure	VSP_<procedure name>
Virtual table	VT_<table name>
Web service	WS_<web service name>

Workflow Object Naming Conventions

A workflow object is an event, task, or gateway. The following table contains standardized naming conventions for workflow objects:

Workflow Object	Suggested Naming Convention
Assignment task	AST_<description>
Command task	CMT_<description>
Exclusive gateway task	EXG_<description>

Workflow Object	Suggested Naming Convention
Human task	HT_<exception table name>_<description>
Mapping task	MT_<description>
Notification task	NTF_<description>
Workflow	WF_<workflow name>

INDEX

A

Address Validator transformation
  execution instances 287
aggregation type
  mapping outputs 101
Aggregator transformation
  cache partitioning 282
  multiple cache directories 283
  partitioned 282
application element
  parameter files 62

B

binding
  mapplet outputs to mappings 111, 113
binding parameters
  instance values 75
  using parameter hierarchy 65
branch pruning optimization
  description 242

C

cache directory
  multiple directories 283
  optimizing 283
cache partitioning
  cache size 283
  description 282
cache size
  partitions 283
commit interval
  target 26
compression codec
  parameters in 86
compression format
  parameters in 86
concurrent read partitioning
  description 273
constraint
  creating 39
  delete row 28
  insert row 28
  target load order 28
  update row 28
cost-based optimization
  description 243
creating the target at run time
  rules and guidelines 136
CUME function
  partitioning restrictions 284
custom mapping configuration
  create 34
custom query
  parameters in 88

D

data object
  get columns at run time 129
data object operations
  described 15
database connections
  pushdown compatible 255
dataship-join optimization
  description 243
date parameters
  valid formats 52
Decision transformation
  disable partitioning 284
  partitioned 282
default mapping configuration
  create 34
development
  mappings 38
dynamic and generated port
  configuration 139
    dynamic mappings
      transformations 139
dynamic expression
  creating 169
dynamic expressions
  example 140
  expression parameters 141
  overview 140
  parameter 141
dynamic mapping
  reorder generated ports 148
dynamic mapping components
  data sources 125
  rules 127
dynamic mapping rules
  overview 127
dynamic mappings
  configuration overview 125
  design-time links 153
  include all remaining ports 144
  include or exclude ports 143
  input rules 142, 144
  link resolution 155
  paramaterize the source data object 131
  parameterize the source name 130, 131
  parameterize the target data object 137
  parameters 127
  port selector 152
  rename generated ports 145

dynamic mappings *(continued)*
configuring Write transformations 172
creating dynamic ports 163
creating run-time links 176
defining input rules 164
developing and running 160
dynamic ports 138
generated ports 138
input rules 142, 144
overview 124
parameterize the target name 137
ports and links 126
restore source port names 147
run-time links 155, 156
running 179
selection rules 152
sources 128
target objects 132
troubleshooting 158
validating 178
dynamic ports
configuring 164
creating 163
overview 138
dynamic sources
descriptions 128
get columns at run time 129
validating 178
dynamic targets
create or replace at run time 135
define based on data object 135
define based on mapping flow 134
get columns at run time 134
validating 178

E

early projection optimization
description 241
early selection optimization
description 242
error
reader 27
transformation 27
writer 27
example
dynamic expression 140
rename generated ports 145
reorder generated ports 149
run-time links 157
selection rules for dynamic mapping 153
example:
expression parameters
in a filter condition 92
in an Expression transformation 93
in dynamic expressions 141
nesting
expression parameters 94
execution environment
Hadoop 24, 25
validation environment 24
Hadoop 24, 25
execution instances
Address Validator transformation 287
Match transformation 287
export
to PowerCenter 212
export to PowerCenter
exporting objects 215
options 214
overview 212
parameter conversion 214
release compatibility 213
restrictions 216
rules and guidelines 218
setting the compatibility level 213
troubleshooting 219
expose as mapping parameter
description 54, 55
task description 74
expression parameter
examples 92
filter condition 92
Filter transformation 92
expression parameters
description 91
Expression transformation 93
filter conditions 92
nesting 92, 94
transformations 92
Expression transformation
dynamic expression 140
Mapping Outputs view 102
partitioned 282
expressions
using parameters 89

F

filter condition
parameters in 97
flat file data objects
reject files 209
flat file delimiters
using parameter 95
flat file sources
partitioned 273
flat file targets
merging partitions 277
multiple output directories 277
partitioned 276
reject files 209
full optimization level
description 245
functions
available in Cloud data warehouse applications 259
available in database warehouse applications 256, 262
available in enterprise applications 265
pushdown optimization 256, 259, 262, 265

G

generated mapplets
creating 44
overview 44
rules and guidelines 44
validation errors 44
generated ports
overview 138
renaming 164
reordering 164

H

Hadoop
  execution environment 25
  reject file directory 25
Hive sources
  parameters in SQL queries 87
how to
  bind mapplet outputs to mappings 113
  configure parameters 68
  define an output expression in mapplet 114
  define mapplet outputs 114

I

IBM DB2 for LUW sources
  partitioned 274
IBM DB2 for LUW targets
  partitioned 280
IBM DB2 sources
  pushdown optimization 254
import from PowerCenter
  conflict resolution 223
  import performance 239
  import restrictions 236
  importing objects 235
  options 233
  overview 220
  parameter conversion 231
  system-defined parameters 232
  Transformation type conversion 224
infacmd
  using parameter sets 81
input linkset parameters
  description 99
input rules
  configuration 142
  include all remaining ports 144
  include or exclude ports 143
  rename generated ports 145
  defining 164
  overview 142
  restore source port names 147
instance value
  setting for parameters 54
instance values
  setting for parameters 75

J

Java transformation
  disable partitioning 284
  partitioned 282
join condition
  parameters in 97
Joiner transformation
  cache partitioning 282
  multiple cache directories 283
  partitioned 282

L

link resolution
  dynamic mappings 155
load order
  constraints 28
logical data objects
  mapping outputs 113
  using parameters 57
Lookup transformation
  cache partitioning 282

M

mapping
  from an SQL query 117
  parameters 53
Mapping Administration
  overview 206
mapping configuration
  create 34
  properties 32
mapping impersonation
  user name 27
mapping outputs
  aggregation types 101
  binding to mapplets 109
  defining 101
  logical data objects 113
  output expressions 102
  Outputs view 101
  overview 100
  system-defined 100
  user-defined 101
mapping parameters
  in virtual table mappings 58
  infacmd 81
  overview 49
  system 50
  types 51
  user-defined 51
  where to assign 82
  where to create 53
mapping pipelines
  description 270
mapping segments
  described 21
mapping views
  described 21
mappings
  adding objects 38
  advanced options 35, 37
  connecting objects 39
  connection validation 23
  creating 38
  creating constraints 39
  decreasing parallelism 285
  development 38
  expression validation 23
  impersonation 27
  mapping configuration 32
  mapping configurations 31
  maximum parallelism 271
  object validation 23
  objects 15
  optimization methods 241
  optimized mapping 245
  overview 14
  parameter validation 23
  partition points 270
  partitioned 271

mappings *(continued)*
pipelines 270
predicate optimization method 242
processing threads 270
reject files 209
run-time properties 24
running 41
synchronization validation 23
tabs 21
target load order constraints 28
validation 22, 40
view 21
workflow 38
mappings, dynamic
troubleshooting 158
mapplet
parameters
override 67
mapplet outputs
binding to a mapping 111
binding to mapping outputs 113
binding to mappings 109
defining 114
how to bind to mapping 115
mapplet parameters
example 56
mapplets
and rule specifications 46
creating 47
description 20
exporting to PowerCenter 213
generating 44
input 43
output 44
overview 42
rules 48
types 42
using parameters with 55
validating 48
Match transformation
execution instances 287
maximum parallelism
description 271
overriding 286
Microsoft SQL Server sources
pushdown optimization 254
pushdown optimization 254
minimal optimization level
description 245
MOVINGAVG function
partitioning restrictions 284
MOVINGSUM function
partitioning restrictions 284

N

naming conventions
object type 292
transformations 290
nesting
expression parameters 94
non-native
execution environment 24
validation environment 24
non-reusable transformation
override
parameters 67
non-reusable transformation *(continued)*
parameters
override 67
nonrelational sources
pushdown optimization 254
normal optimization level
description 245

O

objects
naming conventions 292
operators
available in databases 255
pushdown optimization 255
optimization
branch pruning optimization method 242
cost-based optimization method 243
dataship-join optimization method 243
early projection optimization method 241
early selection optimization method 242
mapping performance methods 241
semi-join optimization method 244
optimization levels
description 245
Oracle sources
pushdown optimization 254
partitioned 274
order
maintain in partitioned mapping 284
stable sort in partitioned mapping 285
ouput expression
configuring in a mapplet 114
output file directory
multiple directories 277
optimizing 277
Outputs Binding
dialog box description 111
Outputs view
description 101
override
transformation parameters 67
overriding
parameters 65
overriding parameters
in mappings 66
using parameter hierarchy 65

P

parallelism
decreasing for mapping 285
maximum 26, 27
suggested 27
parameter
port expression 91
parameter file
rules 62
run mapping 81
tips 62
parameter files
application element 62
creating 64
exporting from Developer tool 64
project element 61
purpose 60

parameter files *(continued)*
running mappings with 60
sample 63
structure 60
XML schema definition 60
parameter hierarchy 65
parameter instance value
setting 54
parameter sets
creating 76
overview 59
parameters
dynamic mappings 127
flat file sources 130
relational table properties 131
source data object 131
target data object 137
exporting to PowerCenter 214
flat file delimiter 95
how to configure 68
importing from PowerCenter 231
in compression codec 86
in compression format 86
in custom query 88
in expressions 89
in filter condition 88, 97
in join condition 97
in mapplets 55
in sql statements 99
in SQL statements 96
instance values 75
logical data objects 57
table names and resources 96
virtual table mappings 58
partition points
description 270
partitioning
Address Validator transformation 287
Aggregator transformation 282
cache 282
cache size 283
concurrent read 273
Decision transformation 282
decreasing for mapping 285
Expression transformation 282
flat file sources 273
flat file targets 276
IBM DB2 for LUW sources 274
IBM DB2 for LUW targets 280
Java transformation 282
Joiner transformation 282
maintain row order 284
maintain stable sort order 285
mappings 271
Match transformation 287
maximum parallelism 271
merged file targets 277
Oracle sources 274
Rank transformation 282
reject files 209
relational connection types 275, 281
SQL transformation 282
transformations 281
troubleshooting 289
partitioning restrictions
Aggregator transformation 282
Decision transformation 282
Expression transformation 282
partitioning restrictions *(continued)*
Java transformation 282
Joiner transformation 282
numeric functions 284
Rank transformation 282
relational sources 275, 276
relational targets 281
SQL transformation 282
performance tuning
branch pruning optimization method 242
cost-based optimization method 243
dataship-join optimization method 243
early projection optimization method 241
early selection optimization method 242
optimization levels 245
optimization methods 241
predicate optimization method 242
pushdown optimization 248
semi-join optimization method 244
pipeline stages
description 270
port list parameters
description 99
port preview
port selector 151
port selector
selection rules 151, 152
creating 168
description 151
in dynamic expressions 140
selection rules 151, 152
port selectors
example 153
ports
connection validation 23
processing threads
mappings 270
project element
parameter files 61
pushdown compatibility
description 255
requirements 255
pushdown optimization
Relational sources 252
SAP sources 255
functions 256, 259, 262, 265
Greenplum sources 254
IBM DB2 sources 254
Microsoft SQL Server sources 254
nonrelational sources on z/OS 254
ODBC sources 254
operators 255
Oracle sources 254
overview 248
pushdwn type 249
relational sources 254
SAP HANA sources 254
sources 252
Sybase ASE sources 254
pushdown optimization method
configuring pushdown 250
full pushdown 249
source pushdown 250

R

Rank transformation
  cache partitioning 282
  multiple cache directories 283
  partitioned 282
reject file
  column indicators 211
  row indicators 210
reject file directory
  on Hadoop 25
reject files
  locating 209
  partitioning 209
  reading 209
  targets 209
  viewing 209
relational data objects
  partitioned 274, 280
  reject files 209
relational sources
  partitioned 274
  partitioning restrictions 275, 276
  pushdown optimization 254
relational targets
  partitioned 280
  partitioning restrictions 281
  reject files 209
rename generated ports
  example 145
reorder generated ports
  dynamic mapping 148
  example 149
reusable mapping configuration
  create 34
reusable transformation
  parameters 67
row indicators
  reject file 210
rule specifications 46
rules and guidelines
  creating target at run time 136
run a mapping
    with parameters
      Developer tool 78
run mapping
    with parameters
      command line 81
      Developer tool 80
      infacmd 81
run-time links
  dynamic mappings 156
  example 157
  link policy 156
  creating 176
  overview 155

S

SAP HANA sources
  pushdown optimization 254
SAP sources
  pushdown optimization 255
scope
  port selector 151
segments
  copying 21
segments *(continued)*
  in a mapping 21
selection criteria
  port selector 151
selection rules
  dynamic mappings 152
  example 153
  port selectors 151
semi-join optimization
  description 244
sort list parameters
  description 99
sort order
  maintain in partitioned mapping 284
Sorter transformation
  cache partitioning 282
sources
  partitioned flat file 273
  partitioned relational 274
specify by
  value or parameter 91
sql statements
  tips 99
SQL statements
  parameters in 96
  parameters with Hive 87
SQL transformation
  disable partitioning 284
  partitioned 282
stable sort order
  maintain in partitioned mapping 285
stop on errors
  mapping 27
string parameters
  in sql statements 98
  precision limit 51
suggested parallelism
  transformations 286
Sybase ASE sources
  pushdown optimization 254
system-defined parameters
  importing 232

T

target load order constraints
  create 39
  description 28
  example 29
  rules and guidelines 29
targets
  merge file 277
  partitioned flat file 276
  partitioned relational 280
threads
  processing mappings 270
transformation
  suggested parallelism 27
transformations
  described 20
  naming conventions 290
  partitioned 281
  suggested parallelism 286
troubleshooting
  dynamic mappings 158
  exporting objects to PowerCenter 219

U

user name
  mapping impersonation 27

V

validate mapping
    with parameters
      prerequisites 79
validation environment
  Hadoop 24
validation environment *(continued)*
  native execution environment 24
virtual table mappings
  configuring parameters 58

W

Write transformation
  dynamic 134